Robust Sentence Analysis and Habitability

David James Trawick

Computer Science Department
California Institute of Technology

5074:TR:83

Robust Sentence Analysis and Habitability

Thesis by

David James Trawick

5074:TR:83

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Computer Science Department

California Institute of Technology

Pasadena, California

1983

(Submitted February 17, 1983)

Acknowledgements

There are many people that I would like to thank for all their contributions that have influenced this work. Unfortunately, I cannot mention everyone personally here, but all of you should know that you are much appreciated.

My first thanks go to my adviser, Dr. Frederick Thompson. Even before he knew me very well, he was able to suggest a course of research that was particularly appropriate for my interests. My work in this thesis has been greatly influenced by his guidance and experience. His encouragement along the way and enthusiasm when things started working has also been a major contribution.

I have been most fortunate in having the influence and help of virtually a second adviser in Dr. Bozena Thompson. She has kept me focused on the central issues of the language phenomena within my area of research. Participating in her experimental research has helped to explicitly define this area. Interaction with her has always increased my facination with languages.

The other members of the sociolinguistics class of '79-'80 deserve many thanks for their work in the experimental analysis. Margaret Cox, Cathy Marshall, and Kelly Roach have all contributed ideas to the results of the experimental research.

A great deal of influence has been made on this work from my associations with my roommate and officemates. I have benefited much from the exposure to other cultures they have given to me. Many thanks to Young-il Choo, Tai-Ping Ho, and Alexandros Papachristidis.

I would like to also thank the other members of the research team for their friendship throughout my stay at Caltech.

Thanks are also due to my friends at Rose Drive Friends Church and the US Center for World Mission. Your encouragement and prayers have been very helpful in keeping me going and in remembering what is important.

I am very appreciative of the suppport and encouragement of my parents. They have been fine examples in all ways throughout all of my studies. Their advice and guidance will always be of great value.

This work was accomplished while the writer was a Member of the Technical Staff of the Hughes Aircraft Company, Ground Systems Group, and a holder of a Fellowship under the Hughes Staff Doctoral Fellowship Program. The generosity of this fellowship program has been a great benefit during my years of study. Contacts with the administration of the program and other relationships with Hughes have always given encouragement. Some supplementary support was provided by the Hewlett-Packard Company, Desktop Computer Division, under a contract with the California Institute of Technology.

ABSTRACT

Systems for using subsets of English with computers have progressed much in the area of linguistic coverage of well-formed sentences for a specific task. Some methods have also been devised for the treatment of input that is almost well-formed. Nevertheless, it is still quite easy to stray over the bounds imposed by current natural language systems. Without proper diagnosis, this leads to interactive systems that are not habitable, i.e., systems that are not pleasant to use because they are not able to perform up to the user's expectations.

This thesis presents an overall system for the treatment of several areas normally outside the limits of natural language systems, and for the diagnosis of any input. The system, Robust Sentence Analysis, includes procedures for handling ambiguous input, resolving input with anaphors (e.g. pronouns), making several kinds of major and minor corrections to input, and the interaction of all of these areas. The system does not treat every aspect of these methods of human interaction, but does provide for the more prevalent forms as found in simulations of user interaction in several modes: face-to-face, terminal-to-terminal, and human-to-computer (using a previously implemented natural language system). Thus the system incorporates the most likely forms found in human performance. Diagnostics are designed to lead the user back into the boundaries of the system.

The Robust Sentence Analysis system is implemented as a part of the ASK System, A Simple Knowledgeable System.

Table of Contents

Chapter 0: Introduction

0.1 Habitability

What happens when the designer of a natural language system steps away from the system and lets someone unfamiliar with the intricacies of its design try to do some real work? After the novelty has worn off, will there be any advantages to using the capabilities of natural language? What are the characteristics of an interactive system that make it a good environment to use? These are all questions pertaining to the habitability of a system. The term "habitable" was coined by Watt, who gave a first definition:

> "A 'habitable' language is one in which its users can express themselves without straying over the language's boundaries into unallowed sentences."
>
> [Watt 68, p.338]

Some examples of user interaction with the REL System [Thompson 75] indicate how easy it is to sidestep the boundaries imposed by a natural language system: (explanations are given off to the side in lower case)

```
>HOW LONG IS THE ANCHORAGE?
 INPUT ERROR:  PLEASE RE-ENTER REQUEST  (vocabulary:"long")
>WHAT IS THE LENGTH OF THE ANCHORAGE?
 6744
>WHAT IS IT'S BEAM?
 INPUT ERROR:  PLEASE RE-ENTER REQUEST  (spelling:"it's")
>WHAT IS ITS BEAM?
 1010
>HOW MANY SHIPS HAVE A BEAM GREATER THAN 1000?
 19
>HOW MANY HAVE A BEAM LESS THAN 1000?
 INPUT ERROR:  PLEASE RE-ENTER REQUEST  (syntax:ellipsis).
>HOW MANY SHIPS HAVE A BEAM LESS THAN 1000?
 13
>LIST THE SHIPS WITH WATER.
 NOUN RELATIVE CLAUSE PHRASE IS VACUOUS. (poor diagnostic)
>LIST THE SHIPS THAT HAVE WATER.
 NOUN RELATIVE CLAUSE PHRASE IS VACUOUS. (poor diagnostic)
>LIST THE SHIPS THAT HAVE WATER LOADED.
 INPUT ERROR:  PLEASE RE-ENTER REQUEST  (syntax)
```

```
>HOW MANY SHIPS CONTAIN WATER?
 INPUT ERROR:  PLEASE RE-ENTER REQUEST  (vocabulary:"contain")
>WHAT IS WATER?
 WATER
```

In the several years since the publication of Watt's paper describing the lack of habitability in natural language systems, much progress has been made both on identifying the individual aspects of this problem, and on working toward correct system processing of some of these areas. These areas of research include: advancing the grammatical range of acceptable input, several approaches for the correct handling of abbreviated reference within context, processing of ill-formed (ungrammatical or extra-grammatical, as in [Kwasny 80]) input, and design of appropriate responses to submissions that violate some aspect of the system's requirements for evaluation. All of these continue to be active research topics and have contributed to making natural language systems better environments for the user.

It must be pointed out, as Watt does, that increasing a language's boundaries to this extent will produce subsets of English that are impossibly large (and also impossibly slow). Thus the goal for a habitable system should be to incorporate the most probable forms of input, and where necessary, give appropriate messages on the limitations of the system. Simple interactive systems do just this; they provide only those constructs necessary for exercising the capabilities of the program, sometimes making the form of input similarly terse. These are often quite habitable, because no one expects them to do anything other than the list of commands they provide.

0.1.1 The Missing Factors

As a variant of Watt's definition, a habitable system is considered to be one in which there is a low probability of users trying to express themselves in a manner beyond the language's boundaries, and when exceeding these boundaries the user receives a diagnostic that will lead back into their confines. Thus the tasks of the system designer desiring to improve the habitability of a natural language system are first, to decide on an ordering of the likelihood of input forms; second, to divide the forms into those that will be processed and those that will be diagnosed; third, to prepare algorithms to process the former set of forms; and finally, to prepare algorithms to generate appropriate diagnostics for the latter set of forms.

Notice that this set of steps assumes that the system designer has adequate knowledge of the forms a user may decide to submit to a system. Thus design for improved habitability often consists of adding some of the newly discovered forms of input that were not at first considered probable enough to include. A habitable system is one that includes these "missing factors", that are more likely than one might first expect.

0.1.2 Competence and Performance Theories

To find these missing factors, there must be study of human performance both with systems and with other people. These results are distinguished from theories of competence, or formal theories of what is generally considered to be correct grammar. The relation of competence to performance was first considered in [Chomsky 65]. This "curious and indirect relationship" [Watt 68, p.340] is behind many of the habitability problems of

natural language systems. Theories of competence usually give the most often occurring syntax forms, but not always. One would like the system to accept the more likely forms observed in human interaction, and yet the system should not reject the forms reflecting competence.

0.1.3 General and Specific Habitability

The kinds of input to be expected by natural language systems varies with the subject matter. A chemical formula may have syntactic variations that will not be expected to occur in the application area of the inventory of a furniture showroom. Thus studies of human performance for improved general habitability (forms common to most application areas) will be different from studies for improving the habitability of a specific application area, and different application areas can be expected to require the processing of different forms.

0.2 Robust Sentence Analysis

Robust Sentence Analysis is the name given to the system developed in this thesis for improved general habitability. It consists of a structure for the processing of forms that will not be included in the larger system's grammar. The additional input forms to accept were taken from studies of people interacting about a problem suitable for identifying the general nature of performance as it differs from competence. Correct system processing of these kinds of input is guaranteed by the algorithms within the structure. The forms treated by the system include several types of anaphoral, ambiguous, ungrammatical, and extra-grammatical input. Diagnosis is provided for forms beyond the capabilities of either the

underlying system or the Robust Sentence Analysis section. Robust Sentence Analysis is implemented within the ASK (A Simple Knowledgeable) System [Thompson 82b].

0.3 Outline of Remaining Thesis

The study of human interaction and analysis of forms for improved general habitability is in chapter 1. The capabilities implied by these forms are considered along with the design requirements for habitability in chapter 2. Chapter 3 presents the structure of the Robust Sentence Analysis system. An overview of the larger environment containing the system is in the fourth chapter. Chapter 5 deals with the processing techniques for more than one equally likely interpretations of the same input: ambiguity. The common forms of abbreviated reference, their treatment and place within the overall system, are presented in chapter 6. The system capabilities for modifying input into forms that are acceptable by the grammar are in the seventh chapter. Chapter 8 describes the final diagnostic, given for input remaining outside the system's grasp after trial of all the other procedures, and presents the algorithm for the creation of that diagnostic. The last chapter gives a summary of the work and some concluding remarks.

Chapter 1: Protocol Analysis

1:You have to sit back and listen to this whole tape?
A:Lucky guy, the things you do for a grade. Transcribe.
1:You sure it's worth it, Dave?

1.1 Introduction (Motivation)

There is ultimately only one source of information about natural language systems, and that is the speakers of a natural language. There is also only one source where information about habitable systems may be gathered, and that is observing these speakers acting cooperatively to solve a given problem. However, there are several ways of gaining this information from informants. The designer may use his intuition and supplement this with ad hoc examples, or may observe the interaction and methods of experimental subjects in a problem solving situation. Protocols are the transcripts of the interaction between subjects of such experiments.

Protocol analysis offers several advantages over other methods. Although it might be assumed that design by intuition is the best if the designer is to be the only user, it will always be incomplete and will probably not include all the capabilities that the designer would find useful. Assuming that there will be other users, it becomes necessary for the design to account for their needs. Some correction in design can be done through ad hoc examples, but this approach is clearly limited. Linguistic theories can provide a base for well understood phenomena, and by analyzing protocols, patterns of natural and habitable communication can be observed in an objective context. Language uses may be found that could not have been found any other way. This method will also tell the

relative frequency of the patterns found, which tells the designer where his efforts should be spent to provide the greatest advantage to the users.

Another advantage to be gained by analyzing protocols is the chance to study the use of language that is not in grammatical sentences. This part of the study is the most enlightening to the subject of habitable system design, because these fragments include the natural ways of streamlining interaction and examples of the kind of error processing that takes place.

Study of the kinds of "errors" people make in communicating and the types of responses these errors generate will help in the design of useful and informative error messages and error correction. Classifying and recognizing the types of fragments that can be safely ignored is also of importance to the system designer.

The fragments encountered that are for the purpose of streamlining the interaction between the individuals provide insight on some of the primary differences between the common uses of English and high level "natural language" systems. One of the prevalent criticisms of systems that try to model human interaction is that they are of necessity too verbose [Shneiderman 80]. When it is realized that "people do not naturally speak in sentences" [Chapanis 75, p. 40] but often use their own forms of terse communication, and when most of these forms can be similarly used with natural language systems, this will cease to be a legitimate complaint.

1.2 Experiment Description

A set of experiments was carried out and supervised over the past few years by Dr. B. Thompson of Caltech, using several different tasks for the simulated problem. In the 1979-1980 academic year Dr. Thompson's sociolinguistics class was involved in this experimental research including the subsequent analysis of the generated data. This was a part of the larger project of developing the ASK System. We wanted to see how language was used by people communicating about a problem that involved interaction such as was needed in using a data base.

The experiment involved two parties communicating over the real life task of loading Navy ships, using data from some ships based out of San Diego. Information was available on the kind and amount of cargo to be stowed, the deck sizes and their hatch sizes, and the dimensions of the cargo. This information was split between the two participants so that interaction was necessary for them to solve the problem. Person 1 was given the data about the ship's decks and the type and amount of cargo to be stowed, while Person A was given the data about the cargo items. (1 and A were used because someone complained about being considered a second-class participant when called person 2 in one of the previous year's experiments.) The actual data as given to the participants are shown in appendix 1.

The experiment was also done in three modes of communication. The first was a face-to-face mode, with the subjects face to face, usually across a table but far enough apart so that they must talk to solve their problem. The subjects were given an hour to work on the problem, and the results proved useful even though only one pair was able to

complete it. A tape recorder was placed between them and the conversation recorded. This was later transcribed by secretaries, and corrected by the experimenter. The protocols were then typed with sufficient space for notes and analysis. Eight of these face-to-face protocols were made, and the results are truly enormous. The average length of one of these protocols was 30 pages. Analysis of these protocols could give interesting results for many other applications.

The second mode also involved two people working on the identical problem with the same data, only this time they had to communicate using a program which simulated teletypewriters while they were located in different rooms. This was called the terminal-to-terminal mode. Four of these protocols were generated. The time limits were the same, with a little time used before the alloted hour to understand how the system worked.

Our third mode had only one human participant, with the other subject being the REL System, the Rapidly Extensible Language System developed by Dr. F. Thompson and Dr. B. Thompson at Caltech. The inputs were slightly different, with the handout given to each user shown also in appendix 1, which was a curtailed version of Person 1's data before (only the number and type of items to be stowed.) Several manuals were available to the subjects for reading before using the system: "Loading Ships Using REL", and the manual REL English for the User [Thompson 78]; plus, there was an experimenter available for consultation. After the subject had learned how to use the interactive loading sequence, not much help was necessary. About a half an hour was given to the subjects to familiarize

themselves with the system, and then the experiment was to proceed for an hour and a half, but some stayed longer to try to complete the task. This third mode was called the human-to-computer mode.

It might be argued that we should have done this third mode with the same data as given in the first two modes, with the person taking in one case the part of Person 1 and in another case the part of Person A. The other extreme might have had the human participant with no data, only knowledge that the system has all the data. The first of these possibilities would have given a more thorough comparison of the three modes, while the second would have given a more thorough test of the REL System. Both of these were among the goals of the experiment, and the method chosen was able to give information on both of these areas. A more in depth evaluation of the REL system should include both of these modes for comparison with the other person-to-person modes.

By studying protocols of user interaction with the REL system, very practical results were generated on how people actually use this system. Of particular interest is the number and type of errors, and the methods of terse communication that were tried by the subjects. But care must be taken in making any generalizations about how people actually use such systems from viewing these results. We can only say how people actually use the system under evaluation. Any other conclusions are influenced by the capabilities of the tested system. This is why the errors are so important. They tell us what was expected of the system that could not be done. But they do not tell the frequency of use of these constructs as they would appear in a

setting where they would be correctly processed. This is why the other modes are necessary for habitability design. Nevertheless, the frequencies of errors are important in terms of how strongly the users would like to have such capabilities available.

The design and guidance of this experimental research is due to Dr. B. Thompson. The experience with earlier versions and a different setting of the experiment has greatly influenced the results here. Dr. Thompson's qualifications as a linguist and familiarity with the REL System from its inception helped the team avoid results that could otherwise have been bogus either linguistically or computationally. The definitive report on this research was published in the paper [Thompson 80].

1.3 Results: Explanation of Categories, with Samples

In order to make some sense out of all the results that were generated, we need to find a way of classifing the elements of communication that were observed. The natural breaks in conversation, indicated by various methods in the various modes, divide the protocols into these elements that we want to classify. These breaks include: pauses, changes of speaker, changes in tone, punctuation and submission of an input to either the teletypewriter program or the REL system. This section will describe the categories that were developed from observation of the protocols, which will include the larger divisions of sentences and fragments.

Types of messages and dialogue patterns [Marshall 80] will also be discussed. These are higher level concepts and provide some insight as to how people generally structure conversations. A message is the term that will be used to

indicate any speech or typewritten communication that has been sent by one individual, without interruption by the other participant. Examples of all the types of fragments in context will be seen in the examples of types of messages, which will be given after the definition and discussion of fragment types. All the examples in this chapter (including the one at the opening) are from the actual protocols that were generated in the experiments.

1.3.1 Sentences

What is a sentence? For our analysis, we took sentences to be the strings that were grammatically sentences as determined by the individual experimenters. All of the experimenters had had experience with transformational grammar, and the notion of sentence in primarily syntactical terms was used to promote consistency among the gathered data. A fact that is perhaps linguistically interesting and certainly comforting for the sake of the experiment was that there was virtually no disagreement as to what constituted the syntactic part of speech sentence. A few exceptions were made to this general rule, which will be discussed in turn.

There were two primary types of sentences considered in gathering statistics in the first two modes. These types were further subdivided in the third mode in order to gain more information on the REL system. The two primary types were (1) the standard sentence including statements, commands and questions; and (2) a variant which almost fit (1) but seemed to be somehow transposed. The first type was like one of the following:

A:As far as we're concerned it's efficiently packed.
A:Why don't we just puree all these things?
1:Put them in a blender.

The transposed sentences are the segments of communication that have all the right parts to be a sentence, but are just out of the standard order, so that they seem slightly transposed. These bring to mind the kind of speech used by the character Yoda in the movie "The Empire Strikes Back" [Kasdan 80]:

A:Inches it says on the front.
1:Those I need nine of.
A:What you need, I can't give them to you.

Both types of sentences were counted as sentences in the statistics for the person-to-person modes.

The exceptions to the general syntactic rule about sentences were whole sentences that make up certain common "phatic" phrases (to be defined specifically later.) Examples of this are phrases such as "you know" and "wait a minute".

The further subdivisions of sentences that were made in the third mode for statistics were division of sentences into commands, statements and questions. Subdivision of each of these was made into categories using pronouns, quantifiers, conjunctions and relative clauses. Each of the questions was further subdivided into classes indicating whether they were wh-questions or not. The transposed sentences that occurred in the human-to-computer mode were not considered as sentences, but were counted as syntax errors, which included both the ideas of extra-grammatical (beyond the grammar of the system) and ungrammatical (outside the grammar of the experimenter) elements. Only the sentences that worked were counted as sentences. Here are several examples of good sentences:

WHAT IS THE REMAINING AREA OF THE MEZZANINE DECK OF THE ALAMO?
LIST THE DECKS OF THE ALAMO.

LIST HEIGHT OF EACH CONTENT WHOSE CLASS IS CLASS IV.
WHAT IS THE SQUARE FOOT CAPACITY OF EACH DECK OF THE ALAMO?
LIST LENGTH, WIDTH AND HEIGHT OF EACH CONTENT WHOSE CLASS IS AMMUNITION.

1.3.2 Fragments

Fragments include all those elements of communication that do not count as sentences. This labeling of fragment types has been developed by the sociolinguistics classes at Caltech involved in these experiments, under the guidance of Dr. B. Thompson. The fragment types fit into three categories: sentence fragments, terse fragments and phatic fragments. Sentence fragments are the fragment types Interruption, Truncation, False Start and Completion. For the terse fragments we have Terse Question, Terse Reply, Added Information, Terse Information, Echo and Correction. The phatic fragments are Phatics, Phatic Connectors and Talking to Self.

Because anything that did not parse in the human-to-computer mode was not considered to be a sentence, there were several categories of errors. These are not necessarily fragments, and some would be sentences if the REL system accepted all of English. These are considered as error fragments. Statistics for these fragments were compiled only for the human-to-computer protocols. Error fragments are vocabulary, spelling, syntax, punctuation, transmission, unexpected input, definition format and bug.

1.3.2.1 Phatic Fragments

Phatic communication has in this context a modified notion of Malinowski's [Malinowski 46] term. We have extended this to mean any form of communication that is primarily for keeping open the channel of communication. This may be with one word

responses that do not answer questions, or it may be with full sentences that have acquired a phatic status from their common usage (as in the exceptions to sentence above.) There are three fragment types that have been classified as phatic fragments, due to their phatic nature in communication.

Malinowski's term is very similar to our notion of phatic messages (see below.) This distinction was made because even within phatic communication many of the other types of fragments and also sentences occur. The fragments discussed here are those that generally have no other use than the phatic one.

The first of these fragments that we will discuss is what has been called the simple phatic, or phatic for short. This consists primarily of interjections that are not part of a sentence, but often precede a sentence. Such interjections include: ok, well, yeah, um, uh, and all the so called four letter words. Names are often used as phatics also. Their primary function is to acknowledge reception of a message or to keep the "floor" to one's self. They may also serve to indicate some shading of meaning to the current sentence.

These simple phatics are often clues that differentiate between various sub-dialects, with certain ones going in and out of fashion over the years. Some popular phatics of the past and present are listed here to give a better idea of the notion of a phatic word or phrase: swell, groovy, far out, ok-fine, totally, for sure.

The next fragment type is the phatic connector. Phatic connectors are words used to string sentences or other fragments together into run-on sentences. They are used in a phatic sense, to maintain the "floor" for the speaker between sentences

that could have otherwise been separated. Words that are used as phatic connectors include: and, or, since, because and so.

The final fragment that can be considered phatic is talking to self. This is verbiage that the speaker may or may not have intended the other participant to hear. It is thinking out loud, and may serve only the function of keeping the hearer from talking while the sender is thinking. In this sense it serves a phatic purpose. Talking to self was generally encountered when one of the participants was doing calculations and was talking to himself about this. This fragment occurred (or more correctly, was recorded) only in the face-to-face mode.

1.3.2.2 Terse Fragments

The terse fragments are the elliptical fragments. They consist primarily of noun phrases, but occasionaly have more of the ellipted sentence present. Terse fragments are sometimes differentiated by their use rather than their syntax. In their respective different contexts, the different terse fragments will elicit appropriate responses in the receiver, in spite of their syntactic similarity.

The use of terse fragments is one of the main differences between current natural language systems and human interaction. It is the way people adapt to their problem and streamline conversational interaction. Studying these fragments has given an idea of the general use of elliptical constructions. These fragments cover the areas of elliptic question, answer, modification, confirmation and correction.

The terse question is simply an elliptical question, and the terse reply is just an elliptical reply. Added information is a kind of elliptical modification, an addition of a relative

clause or other modifier to make more explicit a previous statement. Echoes are an elliptical form of confirming the reception of previous information. (Echoes of whole sentences were not counted as echoes.) A correction is what follows a kind of semantic false start; rather than editing the structure or repeating parts of the current sentence, it changes the information that is being given by replacing one of the previous noun phrases that it contradicts.

The other terse fragment is terse information. It is best understood in context. There are several instances in the examples of message types and the extended analysis example in appendix 2. It may be a kind of terse reply to an assumed question, or just a noun phrase prior to a full sentence that serves to set the focus of the sentence.

1.3.2.3 Sentence Fragments

The sentence fragments are the fragments that have more to do with sentences than the noun phrases found in the terse fragments. The sentence fragments are also not used as the phatic fragments are, but often serve to keep the channel of communication open as the sender endeavours to form a complete sentence or the receiver to complete an abandoned sentence. These fragments can be subdivided into abandoning and completing types.

The abandoning types are interruptions, truncations and false starts. (These are the only categories that qualify as actual speech errors.) An interruption is when a speaker is interrupted by the second speaker and abandons his sentence. This refers to the fragment of a sentence that was interrupted, not the one that interrupted. A truncation is when the speaker

abandons his sentence without any provocation. The difference between these two was usually clear, and in places of confusion the tapes could be used to decide which was the proper category. False starts are those fragments left out when a speaker abandons the first start of a sentence and then completes the sentence. This may involve any variety of ways of backing up and the completion may start in the middle of the sentence.

The only completing type of sentence fragment is the completion. A completion is just what the name implies, with the restriction that the completion is not made by the original speaker, unless interrupted by the other. Completions are completions of truncations, or in the rare case, of interruptions.

1.3.2.4 Error Fragments

In the human-to-computer mode the submissions that did not parse and evaluate to completion were considered error fragments and were classified as such even when some would be considered complete sentences in one of the other modes. The categories give some indication as to the reason why the input did not reach completion. Since all of these could be called syntax errors, with the exception of the transmission and bug errors, the category of syntax error was reserved for those errors that did not fit into any other category.

A punctuation error was one that involved only punctuation. The definition format error was any error that occurred when the subject was attempting to add a definition. Unexpected input errors happened when the subject failed to

answer a prompt that expected a particular format, e.g. "There are 120 lines in your answer. How many do you want?".

Vocabulary errors were any errors resulting from the use of a word that was not in the vocabulary, whether or not this word was misspelled. In order for an error to be a spelling error, it must have been a misspelling of a word in the vocabulary.

Several transmission errors were generated due to the fact that in some of the human-to-computer experiments the parity bit on the terminal was set incorrectly. Once the source of these errors was located, they ceased to occur. Errors generated from the user encountering a bug in the system were called bug errors. These returned a different diagnostic from the other errors, and thus could be distinguished from them. Several interesting phatics were recorded when the user hit a more serious bug, i.e. one that would not let him escape, and these were recorded as phatics.

It should be mentioned here that there were several categories of fragments that gave errors but were not considered error fragments, with the phatic already being mentioned. Other fragments that produced errors but were recorded as in the other modes were false starts and truncations. These can be distinguished from each other only in context. If the user continued with a similar sentence it was a false start; if this sentence was abandoned, it was a truncation. There were also two fragments that worked in a reasonable first-attempt manner for which statistics were taken: terse question and terse reply.

1.3.3 Message Types

In a more extensive paper about this same experiment [Marshall 80], messages have been classified into four types in a study of dialogue patterns. These types are data, metalanguage, phatic and mixed. The distinction between the types is the nature of the dialogue. The data-type messages are primarily for the transfer of data. The metalanguage-type is for meta-level discussion, consideration of various ways of approaching the problem. Phatic messages are messages of a social nature, that often serve as transitions between other message types. Thus the phatic type messages serve the phatic function as discussed above. Mixed messages are those messages that are not predominantly any of the other types.

A knowledge of the nature of message types is necessary for the intelligent design of habitable systems. This is because of two conclusions from Marshall's paper. First, that all message types use fragments and the data types have the highest percentage of fragments. Secondly, that the protocols that finished most of the task had the highest percentage of data type messages. From this it may be concluded that habitable systems should provide for the common fragment forms.

Examples are given here of all the message types, showing each of the fragment forms in context. The commentary in curly brackets is the type of fragment.

Abbreviations:

S	Sentence	AI	Added Information
P	Phatic	TI	Terse Information
PC	Phatic Connector	CR	Correction
SF	Talking to Self	I	Interruption
TQ	Terse Question	TN	Truncation
TR	Terse Response	FS	False Start
E	Echo	CM	Completion

METALANGUAGE

A:So, you've got a list of stuff over there.	⟨PC,S⟩
1:That's right. I've got a list of, I've got,	⟨S,FS,TN⟩
A:2 pages.	⟨CM⟩
1:Yeah.	⟨P⟩
A:I've got 3 pages.	⟨S⟩
1:So it won't fit.	⟨PC,S⟩
A:Three pages can't go into two pages.	⟨S⟩
1:Oh yeah, the super deck is. I have a deck here with no minimum clearance. So...	⟨P,S⟩ ⟨S,PC⟩
A:It must be the outside.	⟨S⟩
1:Yeah, that's obvious.	⟨P,S⟩
A:As long as you don't go above the mast. Perhaps we should use that for the largest objects and then we could stack thousands of little objects in those large ones.	⟨AI⟩ ⟨S⟩
1:I've got an idea. Why don't we just put everything -- you know -- cover the floor with super deck and stack everything on top of it -- we're done.	⟨S⟩ ⟨TN,P,S⟩ ⟨S⟩
A:After we stack the pyrotechnics thing	⟨I⟩
1:Let's go on and use a different approach.	⟨S⟩
A:3932 minus 4 times ... here, do you want to hand the calculator over here?	⟨SF,FS,S⟩

PHATIC

1:Yes, I know I'm making a mess.	⟨P,S⟩
A:He must have learned to eat in Fleming House. That's all you can expect.	⟨S⟩ ⟨S⟩
1:Andy, you have a job to do. Keep filling up that deck.	⟨P,S,S⟩

DATA

1:The Alamo, huh, where are we going with this boat?	⟨TI,P,S⟩
A:Just ship them to, uh,	⟨TN,P⟩
1:Remember the Alamo?	⟨S⟩
A:OK, how about a, how about utility truck M676?	⟨P,FS,TQ⟩
1:I need two of those.	⟨S⟩
A:You need two. All right, one thing left	⟨S,P,AI⟩
1:Only one vehicle left?	⟨TQ⟩
A:One vehicle left and that's a trailer utility F2W.	⟨E,PC,S⟩
1:Trailer utility? I've got a truck utility F2W.	⟨E,S⟩
A:Oh, well, that's different. TRLR F2W.	⟨P,P,S,AI⟩
A:Two without.	⟨TI⟩
1:Yup.	⟨TR⟩
A:One semi-trailer.	⟨TI⟩
1:Semi-trailer, yup one.	⟨E,P,TR⟩
A:10 holister 8 inch 5 tractors M46 with RUP.	⟨TI⟩
1:M46, uh, M64, you mean.	⟨E,P,CR,P⟩
A:Ah, yeah, you're right. I wrote it down wrong.	⟨P,P,S,S⟩

1.4 Statistical Results

The primary objective of the experiment from the point of view of habitable system design was to categorize the observed utterance fragments and determine their relative frequencies. Now that the categories have been described, the number of occurrences and percentages will be given below in tables 1.1-1.8. Table 1.1 shows the breakdown into sentences and fragments. Definitions in the human-to-computer mode are also included in this table. In table 1.2, the relative frequencies are given for the fragment types. Tables 1.3-1.6 give the data for phatic, terse, sentence and error fragments. Breakdowns of the sentences in the human-to-computer mode are presented in tables 1.7 and 1.8.

Table 1.1: Sentences and Fragments

Mode	F/F	T/T	H/C
Total	8763	783	1610
Sentences	3460 39%	382 49%	876 54%
Definitions			55 3%
Fragments	5303 61%	401 51%	679 42%

Table 1.2: Fragment Types

Mode	F/F	T/T	H/C
Total	5303	401	679
Phatic	3467 65%	182 45%	45 7%
Terse	1250 24%	186 46%	151 22%
Sentence	586 11%	33 8%	30 4%
Error			453 67%

Table 1.3: Phatic Fragments

Mode	F/F	T/T	H/C
Total	3467	182	45
Phatics	2714 78%	144 79%	45 100%
Phatic Connector	676 20%	38 21%	
Talking-to-Self	77 2%		

Table 1.4: Terse Fragments

Mode	F/F		T/T		H/C	
Total	1250		186		151	
Terse Response	347	28%	64	34%	84	56%
Terse Question	264	21%	35	19%	67	44%
Terse Information	123	10%	51	27%		
Added Information	271	22%	27	15%		
Correction	8	1%				
Echo	237	19%	9	5%		

Table 1.5: Sentence Fragments

Mode	F/F		T/T		H/C	
Total	586		33		30	
False Start	283	48%	22	67%	25	83%
Truncation	252	43%	10	30%	5	17%
Completion	30	5%	1	3%		
Interruption	21	4%				

Table 1.6: Error Fragments

Mode	H/C	
Total	453	
Vocabulary	153	34%
Punctuation	79	17%
Syntax	69	15%
Spelling	66	15%
Transmission	33	7%
Definition Format	25	6%
Unexpected Input	14	3%
Bug	14	3%

Table 1.7: Sentences (by Complexity)

Mode	H/C	
Total	876	
Simple	668	76.3%
Conjunction	91	10.4%
Quantifier	62	7.1%
Pronoun	28	3.2%
Quant. and Conj.	18	2.1%
Relative Clause	7	.8%
Quant. and Rel. C.	1	.1%
Quant.,Conj.and R.C.	1	.1%

Table 1.8: Sentences (by Format)

Mode	H/C	
Total	876	
Wh-Questions	650	74%
Commands	168	19%
Statements	47	5%
Questions (non-Wh)	11	1%

The primary implications of these results to habitable natural language systems design are that these are the kind of inputs to expect, the kind of outputs to model, and the kind of errors to prepare to correct. The specific details of what this has meant to ASK System design will be discussed in chapter 3, after describing the other goals relating to improved habitability. In the rest of this chapter several other experiments involving protocol analysis and their application to this work will be discussed.

1.5 Other Protocol Analysis

It is fortunate that there are several reports on experiments involving protocol analysis in the literature. Comparison with this work will provide the benefit of other perspectives on this part of system design. The work in this area breaks down into four major divisions, each with differing goals and each with something to contribute to habitable natural language system design. These four divisions are: other reports on the same experiment described in this chapter, various simulations of natural language systems, earlier linguistic categorizations based on protocol analysis, and previous evaluations of existing natural language systems.

1.5.1 Same Experiment

Much more has been written about the same experiment reported on here in the papers [Thompson 80] and [Marshall 80]. The first of these papers is the primary report on this experimental research, and contains a wealth of analysis and the germ of many of the ideas developed in this thesis. The results

of the second paper and the important conclusions about message types have been discussed earlier.

Phatics words were first described in [Marshall 76], a report on an early version of this experiment. Four types of phatics are distinguished with varying shades of meaning. These types are typified by the examples ok, well, wait and no, meaning tacit acceptance, doubtful acceptance, asking for time and a preface to contradiction, respectively. These different uses dictate how phatic fragments should be viewed by the system.

In a forthcoming paper [Thompson 83], the same experiment has been performed for analysis of the use of pronouns. Statistics have been gathered on noun phrases, pronouns and their relations. This study gives important values for the initiation and updating of the degree of activatedness of a noun phrase, which will be called the salience. These values will be discussed in the chapter on anaphora.

1.5.2 Simulations

There are several simulations of interactive human communication for the purpose of improved interaction with computers reported about in the literature. In an introductory paper [Chapanis 75], several different communication modes were considered, with the primary result being the dependence of the time of completion on the mode. The time increases as fewer means of communication are available. This shows that the addition of other communication forms will aid the user, such as graphics. The analysis of this experimental research is expanded in the reports [Chapanis 72] and [Chapanis 77].

The dependence of task completion on the limitations of vocabulary was the topic in [Kelly 77], with the major result being that the subjects who worked with restricted vocabularies solved their problems as successfully as those with no restrictions. However, frustration and occasional anger was prevalent among the participants working with limited vocabularies, showing at the very least the need for system adaptation to users' idiosyncrasies.

An extensive simulation for design of a specific system is the subject of [Malhotra 75]. The results here confirm the fragmentary nature of human interaction, and provide design requirements for the application area of a management system. This paper is an example of the kind of study that should be made to find out what capabilities users would desire in a system for a particular domain.

Another simulation comparing several different modes of communication is [Cohen 82]. The analysis is in terms of the (task) function of the utterances, as opposed to the concentration on utterance form in our experiment. The primary result is to highlight the differences between speech and typewritten interaction, thereby showing the needs of speech processing systems that are beyond the needs of typed interaction.

The emphasis on an utterance's task function makes this study less applicable to general system design, although it is more applicable to the design of systems for the simulated problem (assembling a water pump, which might be generalized to assembly procedures.) The emphasis on form in this thesis and in [Thompson 80] has highlighted the similarities between speech

and typed communication, even when some of our forms have been differentiated by (dialogue) function (e.g., all of the terse fragments are often elliptical noun phrase fragments but occur in different locations within a dialogue, each performing a differing dialogue function.) Our emphasis on form helps to determine the dialogue function of a given fragment, rather than the task function, so that the nature of human interaction can be studied rather than interaction about a specific area. Both of these areas are important to the overall habitability of a system, as will be seen in the discussion of evaluations below. Interaction of a general nature has been the emphasis here in order that a habitable base for any applications area may be designed.

1.5.3 Categorizations

There have been several linguistic studies involving protocol analysis that have yielded various categorizations of the non-sentence utterances prevalent in speech. In an extensive analysis to categorize words by the positions they may assume in sentences, Fries [Fries 53] confirms several of the notions of phatic words. In [Bowman 57], minor and fragmentary sentences are studied and most of the types of fragments as described here can be found, with some interesting results on the dependency of minor sentences on major (full) sentences. The split is roughly half and half between dependent and independent minor sentences. This says that of the minor sentences (including elliptical forms), some may not have a context for expansion to a full sentence.

1.5.4 System Evaluations

Evaluations of natural language systems will usually produce a list of statements submitted by users as at least part of the evaluation, which is a protocol available for analysis. A long list of submissions is included in the appendices of [Damerau 81], which show the utility of the TQA system as applied to the data base of the Planning Department of the City of White Plains, New York. There is a list duplicating the submissions that did not reach completion, with the ones that worked after a revision of the system marked. This revision, prompted by the viewing of the protocols, displays the utility of such analysis as many of the previously non-working user inputs were completed by the later version.

Three major areas of systems evaluation are given in a thesis devoted to this topic [Tennant 80]. Two of these areas involve taking user protocols: habitability and completeness analysis. The other area is abstract analysis, which is an evaluation from the designer's point of view of a more theoretical linguistic nature. The first two areas consider habitability from different angles, one the habitability of interaction and the other the task habitability. This distinction has been seen in the comparison between the results given in the present thesis, concentrating on interaction habitability, and the work of for example [Malhotra 75] and [Cohen 82], which emphasize a particular application and hence task habitability. Tennant's including both of these areas in a general evaluation method shows the need for both types of analysis to achieve true habitability for a given applications area.

1.6 Summary

At the end of this tour of protocol analysis in the given experiment and in the literature, it can be seen that there is much to be gained from this kind of study from many points of view. There is also no end to protocol analysis. It is always a useful tool in designing, testing and evaluating interactive computer systems. It gives the experimenter the valuable hindsight needed to do all of these tasks relating to system development, plus the satisfaction of certain intangibles, such as having paid one's dues.

Chapter 2: Design Requirements for Habitability

2.1 Introduction

This chapter presents a set of goals for design of natural language systems with improved general habitability. Input forms that have been viewed in the protocol analysis are considered from the standpoints of: what non-sentence forms should be accepted as good input, what forms can be appropriately ignored, what forms can be considered erroneous, how they should be corrected and/or diagnosed by the system, what other system actions require explanation, what priority should be assigned to different forms, what forms have equivalent priorities, and what limitations on these considerations are introduced by the mode. These are the capabilities that need improvement in most natural language systems. Both the processing by the system and the use in output messages of the various forms found in protocol analysis will be considered. The overall design for meeting these goals is presented in the next chapter.

Certainly the goal of a habitable natural language system in regard to grammatical sentence input should be to accept any well formed input relating to the knowledge base. This is a major thrust of most of natural language systems research. But grammatical completeness is not the concern here. Even applied to specific domains, natural language has long resisted attempts at formalization. Rather, the lack of grammatical completeness is a strong motivation for this work. Systems should respond in a reasonable manner to extra-grammatical input, which is anything that exceeds the system's grammar. In the absence of grammatical completeness a habitable system should try to maintain a responsive completeness, giving answers that

delineate for the user the limitations of the system when and where they are exceeded. This is the design goal concerning completeness.

A habitable system should also complete each request in a reasonable amount of time. Keeping response time down is a major concern here, and this is kept as a design goal throughout.

2.2 Non-Erroneous Non-Sentence Forms: Terse Fragments

The fragment forms found in the protocol analysis that are not sentences but contribute specifically to the solution of the problem are the terse fragments. These forms of abbreviated reference serve as a natural way to streamline the interaction between individuals. Study of message types shows that the terse fragments predominate in the data message types, where problem solving principally happens. The terse fragments occur more frequently when the participants have agreed on the approach to the problem, and thus the interaction with the terse fragments reflects the choice of this approach by using a sub-language that follows this method.

A habitable system should provide for the specification of such knowledgeable sub-dialogues, instantiating a particular approach to a given problem. Within the ASK system, this topic has been discussed in [Ho 83] and [Thompson 82a]. Transcripts of user interaction with Ho's dialogues will show that most system and user messages are terse fragments (primarily terse questions and responses). Here is an annotated example, with user inputs in lower case:

```
>load                          (terse request)
>SHIP: alamo                   (terse question, terse reply)
>CARGO SPACE: mezz deck        (terse question, terse reply)
>ITEM: culvert                 (terse question, terse reply)
 STOWED.                       (terse reply)
```

Similar analysis of systems using icons, menu selection, form filling, etc., will show the widespread use of systems using terse fragments as a primary means of communication with the user. The naturalness of these interaction methods has prompted several researchers to think that natural language systems are too verbose ([Shneiderman 80].) But the use of terse fragments in the protocols before the explicit declaration of an interaction method shows that these messages must be understood in a general setting, providing a bridge between interaction in full sentences and knowledgeable (of specific tasks) terse dialogues.

It should be conceded that for a specific task that will not change over time a knowledgeable application dialogue is the most natural form of interaction (after any instruction needed for its use.) But for new applications of an existing data base, exploring new techniques for solving an old or new problem, or just perusing data to get ideas of how to go about solving a problem not yet formalized, a natural language geared to the data base accepting both sentences and the terse fragments is the more habitable choice.

How should a system respond to terse fragments encountered outside the context of a previously defined dialogue? Responses to terse questions would be computed by interpreting these questions as elliptical fragments relating to previous input:

```
>who is the mother of John?
 Mary
>Bill?        (i.e., who is the mother of Bill?)
 Joan
```

There should also be appropriate answers made to terse questions outside the context of any previous input, and terse questions that could be interpreted either way. The system should be able

to accept terse responses to any questions it asks. Corrections will change the right data. Terse information will make the right changes in focus, and added information will add its information correctly. Echoes will just repeat the same processing.

A habitable system may also make use of terse forms. The answers to most user questions can be in terse reply form, rather than in complete sentences. There should be no need for corrections. Terse questions, terse information and added information will occur in system dialogues. Echos may be used for diagnostic purposes (see below.)

2.3 Ignorable Forms: Phatic Fragments

The fragments that are for keeping the channel of communication open or for keeping the "floor" are usually not needed for interaction with computers, and when encountered can be safely ignored. This is due to limitations of the setting, discussed below. Because a person will often think about what to say next while another is talking, and because either individual may initiate in the discussion, phatic fragments help keep discussions orderly, at least to an extent. The shades of meaning conveyed by phatic fragments are of a very subtle nature if they are to be truly understood, but fortunately are only a social aspect of problem solving interactions. Thus if phatics and phatic connectors are recognized they can be ignored, and the system will perform correctly whether the user is cursing or praising it. The talking-to-self fragments, while very rare in modes other than speech, demonstrate the need for a system capability for ignoring various portions of input while attempting corrections.

The process of ignoring phatic fragments is considered in the sequel as a type of correction. This is because while ignoring phatics is very close to a correct interpretation of phatics, no attempt was made to really use what little information they might convey. Instead, the input was corrected to a wholly executable input containing no phatic fragments. This change in the input is reflected in an echo without the phatics, so the user will not be misled into thinking they have been "understood" (although sometimes this echo may be suppressed):

```
>Well, I really don't want to do that.
 I don't want to delete all files.
 no files deleted.
```

Some use of phatics by the system may be worthwhile. Phrases such as "thank you" are used by several systems, and politeness should be a part of habitability (but not carried too far.) Other messages such as "ok" are an appropriate response to requests for output that are not verbal (or typewritten) such as plots or graphics, either before or after the request has been completed (but after language processing.) For questions or a system that takes very long to process input (30 seconds is very long if you are used to dealing with people), some phatics to indicate continued system processing would keep the user from thinking that it will never return.

2.4 Erroneous Forms: Sentence and Error Fragments

Several forms were identified in the protocols which are really fragments of abandoned sentences. These are the truncation, false start and interruption forms. The understanding of these forms and even their definition requires at least a guess as to what the abandoned sentence was.

Completion fragments also require knowledge of this implied sentence. System acceptance of these forms thus requires finding the full sentence, which is considered to be correction of the input.

The sentence around a false start may be found by selectively deleting parts of the false start. Truncation/completion pairs will give the implied sentence, and user interruptions by the system should not take place. (Of course, the user may interrupt the system anytime.) Otherwise, only some kind of expectations or inference can help return to the abandoned sentence. Fortunately, finding the abandoned sentence is not really necessary if it was abandoned by the user. But since abandonment is even more difficult to determine from the system's point of view, attempting to find the abanboned sentence for execution is a recommended solution for improved habitability, since the system can expect to encounter some of these forms. Otherwise, the part of the input that does work may be accepted:

```
>what is the length of culvert, post 2ft, and
 what is the length of culvert, post 2ft?
 culvert    48 inches
 post 2ft   50 inches
```

The error types identified in experimentation with the REL system give a list to the system designer of common input errors that should be corrected. Actual methods of error correction for vocabulary, punctuation, syntax and spelling errors will be discussed in the algorithm descriptions given in later chapters. All errors should be diagnosed. Definition format errors should only be diagnosed, because additions should not be made to the system based on system guesses. Definitions could be made more easily by allowing several formats, thus improving

habitability. Unexpected input errors are corrected by having the semantics that requested the input accept other forms also. Transmission errors were errors of the experimenter and not the user, and so should not recur. Correction of bugs is the responsibility of the system implementers, but a diagnostic will be helpful to the user for bugs that have been identified but not corrected.

There is no appropriate system use of these forms, except perhaps completion as a diagnostic, but it is more appropriate and clear to the user to echo the entire abandoned sentence if a completion has been found. Habitability is not improved by errors in system responses. Any system with enough capabilities to be interesting will have enough bugs to take care of this if it were desirable anyway.

2.5 Diagnostics: Who Needs Them?

Diagnostics are any messages generated by the system that explain system behavior rather than answer the specific input of the user. A general rule will be adopted here that anything the system does that is not explicitly requested in the input should be explained in a suitable diagnostic. This includes ambiguous, ill-formed, and incomplete input or input requiring the resolution of a reference. Any input that is ambiguous requires diagnosis, so that the user will know how the input was interpreted. Ill-formed input is diagnosed in order that any corrections made and how to make well-formed input in subsequent submissions will be understood by the user. Incomplete and referential inputs need the explanation of what was actually executed, how the references were resolved.

There are other situations commonly occurring in natural language systems that require an explanation of what happened when a user request was executed. A good example is the response to inputs that are semantically vacuous:

>How many students took linguistics 101 in 1981?
A: Nothing in data base.
B: Vacuous noun phrase.
C: There was no linguistics 101 in 1981.

Certainly answer C is more helpful to the user than either of the first two. This situation is studied more fully in [Kaplan 78,79], and in [Trawick 79] appropriate answers are considered as applied to the REL setting (which have been incorporated into ASK system design.) These kinds of diagnostics will be called semantic diagnostics, to distinguish that the input form was correct and no references needed to be resolved. The purpose of a semantic diagnostic is to give a more understandable answer rather than explain what was done. The other kind of diagnostic will be called a syntactic diagnostic, as it seeks to explain what was lacking in the form or specification of the user's input.

2.5.1 Rules for Good Diagnostics

Since so many areas of concern to habitability design need explanation, an understanding of what makes for a good diagnostic is important. Good diagnostics contribute to improved habitability while bad ones detract from system habitability. Here is a list of characteristics that should be true of ideal diagnostics. They do the following for the user:

1)Give help in forming correct input;

2)Help user avoid making similar mistakes;

3)Avoid "computerese" or other technical vocabulary, such as linguistic or system specific terms;

4)Are constructive rather than critical of the user;

5)Do not dehumanize the user or suggest system control over the user;

6)Do not anthropomorphize the system;

7)Do not exceed the system's vocabulary;

8)Use the forms commonly found in protocols that serve a similar function;

9)Are brief;

10)Are correct.

Unfortunately, it is seldom that all of these goals can be met, and in given circumstances some may be contradictory. The design decisions involving the selection of a specific diagnostic are important and will be discussed when they are presented.

Several other admonitions for system message design can be found in [Shneiderman 82], which have influenced the selection of guidelines given here (rules 1-5). Rule 6 reflects a desire to have a professional system rather than a "cute" system, although messages should not appear harsh or cold to the user (rules 3-5). Rule 7 is of particular interest to the setting of natural language systems, because it is reasonable for a user to expect that any word used by the system can be accepted by the system. However, this is often very difficult to do in practice when explaining to a user the nature of the problem with his input.

The form found in the protocols most useful for diagnostics (satisfying rule 8) is the echo. In the protocols, an echo usually serves the function of repeating for confirmation the

last message of the other speaker. Thus a natural language system would do well to use an echo as a diagnostic of what the system did if that was anything other than the (explicit) input, as in the case of errors and references.

Rule 9 is included so that the interaction will not be hindered by too much diagnosis. Perhaps the most important is rule 10, which must be true for a diagnostic to be of any use.

2.6 Priority

Priority questions arise in good input because of ambiguities. (Sentences and the terse fragments are considered to be good input.) Until chosen by the user, each ambiguous interpretation of the input has equal likelihood of being what the user wanted and is given equal priority. Systems that are fast enough may carry out all of these interpretations and give each result to the user, but any habitable system should have messages explaining the differences for the user's selection, either along with the output or before evaluation.

Several design goals apply to the priority of error correction. Because the primary use of any system is with well formed input, extra time should not be spent by correctors operating on good input. Secondly, corrections near the original input should be tried before other more radical corrections are attempted. Thus a check for potentially misspelled words will take place before a change in the input's syntax. Finally, each possible correction at a given level of change away from the original input should be tried before moving on to further levels. This notion of distance dictates the priority that the correctors have. If there is more than

one correction at a given level, then the input is considered to be correctionally ambiguous, giving the corrections equal priority.

2.7 Limitations Due to Setting

This chapter has been more general than was actually necessary for the discussion of ASK System habitability design. Some forms found in the protocols will not occur due to mode limitations and system capabilities. Because ASK is a system using typed input, and because the computer it is implemented on allows a certain amount of editing of an input before submission, false starts are far less likely than they were with the REL system. A type-ahead feature keeps system output from interrupting subsequent user input. Terse responses can be expected only after questions asked by the system, which occur much less often than questions of the user. Phatics seem less necessary without the need to arbitrate between leadership within the problem solving team. However, they are very frequent in the presence of bugs, which amount to a nasty way of describing the system's limitations. (Most phatics prompted by the presence of bugs do not contribute to problem solution, and are probably not fit for repetition in polite company.) Systems that operate in different modes, particularly ones proposing to handle speech as a method of input, will have to give more thought in their design to the fragment forms that are reduced by the typewritten setting.

Chapter 3: Design Structure for Habitability

3.1 An Overall Framework: The Metric, Informally

How can the processing of good, ambiguous and ill-formed input and input requiring resolution of references be combined into a design with more unity than a bag of tricks? All of the seemingly disjoint areas to be covered in this design are brought together by an informal metric to be described here. Complete, good input is input that can be parsed by the system's grammar to the part of speech sentence without any pronouns or other needs for reference resolution. Any other input must be transformed into such a sentence, either through correction or resolution of a type of reference. The nature and number of changes made to a given input to make this transformation gives rise to the idea of distance between input strings. Ambiguities arise whenever an input is equally close to more than one complete, good input.

The metric is never computed and only serves to give intuitive structure to the design. It serves the same purpose that the approximate string matching of the Damerau-Levenshtein metric [Hall 80] does for spelling correction. It shows how an erroneous input or string may be modified in order to first get the corrections nearest to the input and later proceed to corrections further away from the input, making explicit the priorities assigned to input forms. The design of robust sentence analysis is essentially a description of the perceived metric between input strings.

3.2 Design for Robust Sentence Analysis

Without giving the implementation details of specific algorithms the structure and sequence of "corrections" on input are described here. The metric is an extension of the Damerau-Levenshtein metric incorporating a knowledge of the grammar involved and operating on the words of an input string. Thus instead of insertion, deletion, replacement, and interchange of adjacent letters, these operations are on the words, punctuation and other special characters of an input string.

Insertion:

```
>What is the destination of the Maru?      (context)
 London
>of the Alamo?                             (needs insertion)
 What is the destination of the Alamo?
 New York
```

Deletion:

```
>What is the destination fo of the Alamo? (delete "fo")
 What is the destination of the Alamo?
 New York
```

Replacement:

```
>What is the destination of the Alemo?    ("Alamo" for
 What is the destination of the Alamo?        "Alemo")
 New York
```

Interchange:

```
>What the is destination of the Alamo?   ("the" and "is")
 What is the destination of the Alamo?
 New York
```

The value of the distance (and hence the order of correction attempts) is influenced by the part of speech of the words subject to these operations. This is to incorporate a knowledge of the forms found in protocol analysis into the structure.

3.3 Well-Formed Incomplete Input

All input parsing wholly into one part of speech is considered to be well formed, and thus has priority over corrections. Input that contains a pronoun or is not in the part of speech sentence needs resolution of a reference and is considered incomplete because of this. The resolution of a pronoun is the interchange of a pronoun for the noun phrase it stands for. This interchange is the nearest change to the input that can be done. Following this is the expansion of other parts of speech to sentences by the addition of appropriate words from the interaction context. These additions include the filling out of all elliptical fragments.

Noun phrases are a special case because of the goals for treatment of the terse fragments. The part of speech noun phrase parses into a sentence so that stand alone noun phrases can be replied to without expansion into sentences. The context is used to determine which response is appropriate, depending on the complexity of the noun phrase.

3.4 Ill-Formed Input

If the parser fails to find any well-formed input, then error correction takes place. There are several minor corrections that should all be given the same priority. These are the correction of spelling and punctuation errors and the deletion of phatics and phatic connectors. Replacement of a word not in the vocabulary by one that is constitutes a spelling correction. Correction of punctuation is by deletion. The deletion of punctuation and phatic fragments is done in a manner that is optional, so that all good results may be obtained after resubmission to the parser. This is desired

also because phatic words may be lexically ambiguous with other interpretations not being phatic. All well-formed corrections are then processed, which may potentially involve the resolution of references. If a word was found that was not in the vocabulary, a diagnostic to this effect is generated. As a second step optional deletion of words not in the vocabulary may take place. Any correction is diagnosed by echoing the corrected input as it was executed.

If these near corrections fail to produce well-formed corrections or a vocabulary diagnostic, others are invoked. First, several attempts are made to see if replacement of words by other forms of the same word will correct the input. Multiple replacement of certain parts of speech to all words with the same part of speech (such as prepositions) is also included at this stage. These correspond to the feature and part of speech (test and category) relaxations of [Kwasny 80]. A second level spelling corrector, that considers all input words subject to spelling correction is the next furthest modification of the input. Finally, correctors attempting changes in word order and the optional deletion of input words are called. Any correction is diagnosed as before.

If no correction is found and there was no vocabulary message generated, then a diagnostic is generated informing the user of the situation. Another more detailed analysis of the input is available to the user on request if more information is needed to explain the nature of processing as completed on the input. This more detailed diagnostic is an extension of the diagnostic recommended in [Weischedel 80], using the heuristic of the deepest parse. It communicates to the user the fragments

that were identified in the input and retains the full ambiguity of fragment combinations, not being prejudiced by a top down approach. This message will be called the maximal covers analysis diagnostic.

3.5 Ambiguity

The equal treatment of individual ambiguities that are the same distance from complete, well-formed input is maintained throughout the system. If the ambiguities are correctional, then the corrections serve as disambiguating messages, and the syntactic diagnostic precedes this other output. Otherwise, diagnostics are generated explaining the different interpretations of the ambiguity, whether structural, lexical or definitional. All ambiguities are processed through to completion, and are then presented to the user. Some interpretations may produce semantic diagnostics. If there are any ambiguities giving a good result, all of the semantic diagnostics are suppressed. All diagnostics are output if there is no good interpretation.

3.6 Comparison with Other System Designs

Another system design using the notion of a metric between well- and ill-formed input is Bradford's ETA (Error Tolerant Analysis) system [Bradford 82a, 82b]. This considers word deletion, insertion and adjacent interchange in order to find well-formed input. (Substitutions are deletions followed by insertions.) The measure is defined formally. The system uses the grammar to find the closest sentence in the grammar to the input, and presents the alternatives found to the user one at a

time. This proceeds until the correction is found or the user terminates the process.

Weischedel and Sondheimer have a structure for dealing with ill-formed input within the ATN setting by using metarules that change the system's grammar to accomodate correctable errors [Weischedel 81]. This gives an elegant format to the ideas in [Kwasny 80] on rule relaxation, and coupled with Kwasny's pattern arcs would go far toward improved habitability.

Hayes and Reddy have a design for a "graceful" system, which is a design for improved habitability [Hayes 79]. Their paper gives the design goals for a forthcoming system in which they expect to include the speech mode. Their study concentrates on limited settings designated "simple services", such as reservation systems for airlines or restaurants.

These other system designs have been studied and the present design has attempted to include the good features of each and to add many new capabilities to the state of the art. The major differences are the treatment of the fragment forms found in protocol analysis, primarily the phatic and terse fragments, and an organized treatment of ambiguities and diagnostics. In addition, each method of correction and the resolution of references has been carefully studied in terms of response time, generation of all appropriate changes a given distance from the input, and the interaction of the various changes made to the input.

Chapter 4: ASK System Environment

4.1 Introduction

A vocabulary will be necessary in the remainder of this thesis describing the language processing routines available within the ASK System. Execution of these procedures on an input string constitutes an evaluation of that string. The programs for correction of input to that which is well-formed and complete make use of those described here. All of the corrector/diagnostic algorithms to be discussed later are accessed as required by the degree of completeness of the results obtained from the language processing procedures. Thus an understanding of what the language processing routines do is necessary before the explanation of the corrector/diagnostic portion of the system. Furthermore, each of these procedures contributes to the overall system habitability because of their underlying ambiguity handling capabilities.

4.2 Languages in the ASK System

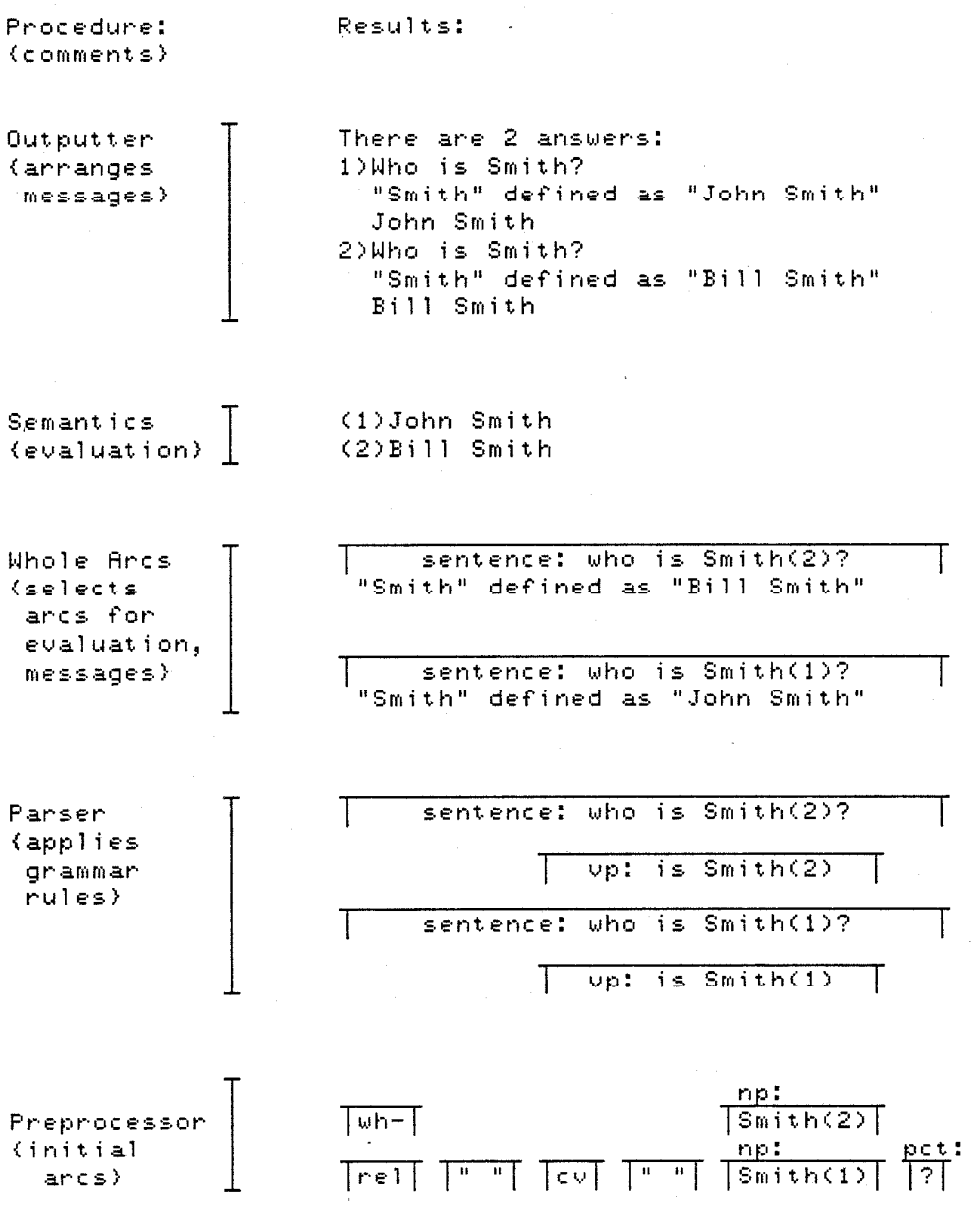
First some description of ASK System philosophy is necessary to understand these descriptions. The underlying system implements a processor for formal languages as defined in [Thompson 66]. Concisely described, this is a general rewrite rule grammar with associated semantic procedures that are invoked as required by the application of the grammar rules. The language processing routines perform functions corresponding to the divisions in this definition. An applications methodology is maintained as described in [Bigelow 73], where individual applications receive their own language. A language consists of a lexicon and a grammar with its associated semantic routines.

Semantic routines are called in the order prescribed by the grammar, and may be executed either at syntax or semantics time as stated in the grammar rule. Input that is syntactically well formed guarantees that the semantic routines corresponding to the grammar rules used will have the right types of input at the time of their execution.

A language is specified in the ASK System framework by grammar rules with associated semantic routines. Individual words are added to the lexicon either by lexical rules in the grammar or by definitions. The language processing routines are independent of any particular language. Base languages are maintained, such as ASK English, for the purpose of building applications languages above them. These basing facilities are described in [Yu 80]. Because of the rapidly changing nature of languages and the possibility of the base languages being other than English (others are presently under construction), language processing, correction and diagnosis must only depend on what can be found in all languages.

4.3 Overview of the Language Processing Environment

The language processing environment of the ASK System consists primarily of five procedures. These are the preprocessor, parser, whole arcs, semantics and outputter routines. The basic function of these programs is illustrated in the following diagram. It indicates how the programs add to the parsing chart, and is presented in a bottom-up manner reflective of the bottom-up nature of the parser. This example is a definitional ambiguity.



Input String: Who is Smith ?

with the parts of speech abbreviated as:

vp: verb-phrase
np: noun-phrase
pct: punctuation
wh-: wh-word (e.g., who, what, when, where, why, how)
rel: relative pronoun (prefaces relative clause)
cv: copula verb

4.3.1 Preprocessor

The preprocessor is the first procedure of the language processors to analyze the input string. It forms the initial parsing chart. This is a list structure that contains the parses of characters, words in the lexicon, identifiers ("words" not found in the lexicon), and real and whole numbers. If a word is in the lexicon with more than one part of speech or with differing features, it is added to the parsing chart ambiguously. Any affixes to words that were included in the lexicon also appear in the initial parsing chart. An example of these are the prefixes for units: milli, pico, kilo, and so on.

4.3.2 Parser

The parser applies the grammar rules to input of the form of the preprocessor's output. This is done in a manner retaining any ambiguities of the input. Each arc in the parsing chart returned by the parser has links to both the syntax and semantics of the arc. The syntax of an arc consists of its part of speech, features and the underlying literal string that has parsed into the arc. The semantics may be a variety of things: pointers to semantic routines, numerical values, free or bound variables, or definitions. The structure is inherent in the pointers connecting the arcs. Any semantic routines to be executed at syntax time happens during the application of the corresponding grammar rules. All of this structure is available after the parser's execution for analysis by other routines, which can then determine the status of the input's well-formedness.

4.3.3 Whole Arcs

The whole arcs procedure searches the output of the parser for any arcs covering the whole input. Preference is given to arcs with the part of speech sentence, so that if any sentences are found, only the sentences are kept. If no sentences are found, all arcs covering the whole input are kept. If no arc is found that covers all the input, then the search is redone to look for arcs not covering the trailing punctuation. This is because sentences include the end punctuation (optionally), while other parts of speech do not. Elliptical fragments with trailing punctuation will thus be found. In the case that there are still no whole arcs found after relaxing the requirement to cover the final punctuation, a nil list is returned.

4.3.4 Ambiguity and Anaphora

If the whole arcs program finds more than one arc to retain for further processing, then diagnostics must be prepared explaining the interpretations found. The routines for the generation of disambiguating messages are called in this case. The procedures for the resolution of references are called after finding the whole arcs so that the syntactic relationship between the elements of the input may be used for this resolution and the updating of the potential referent list.

Another reason for anaphora processing at this time is so that reference resolution may take place in conjunction with other corrections. This remark also applies to the disambiguation procedures. They are called whenever there is more than one good whole arc so that they may diagnose both true and correctional ambiguities.

4.3.5 Semantics

Any remaining semantic routines are executed by the "sem" program. This receives a list of whole arcs as input and calls the semantic routines as indicated in the parsing chart. It returns a list of phrases prepared for the outputter called OUT phrases. If there are no good results at the end of semantic processing, this list is nil.

4.3.6 Outputter

A list of OUT phrases with disambiguating messages is sent to the outputter as the last step. The outputter takes this information and generates a list of lines for output. The line list made contains all good outputs if there are any, and otherwise has all semantic diagnostics. Any syntactic diagnostic is always included at the begining of the list. All interpretations (disambiguation messages) giving the same answer are listed before that answer. If the line list is longer than what is considered reasonable (usually 20 lines), the user is prompted as to how much of the output is desired. If there are more than three ambiguities, a different prompt is given along with the interpretations for user selection. The user may always see the rest of the line list, any other line range in the line list, or the other ambiguities that were not requested in the first response.

4.4 Diagnosis and Correction: How and When

Various failures of the language processing procedures will prompt execution of correction and diagnostic programs. Failure of the preprocessor only happens on a nil input string, and the input of nothing is given the appropriate response of nothing.

Parser failure invokes the program administrating the correction and diagnostic routines. The goal of these routines is to either get good results through the semantics stage or to at least give a good diagnostic. Each level of corrections consists of calls to all of the language processor routines other than the outputter interspersed with the execution of the modification programs. Higher levels of correction are performed upon the failure of the lower level correctors. In this manner, the input is reevaluated with changes successively more radical in nature.

Chapter 5: Ambiguity

> Jane:I know a man with a wooden leg named Smith.
> Michael:What's the name of his other leg?
> (from "Mary Poppins" [Walsh 64])

5.1 Introduction

Ambiguity arises whenever input has more than one equally likely interpretation. This may happen with both well-formed and ill-formed input. Well-formed input is ambiguous when the input corresponds to sentences normally considered ambiguous in natural conversation, either due to syntactic, semantic or referential considerations. Ill-formed input is ambiguous if there are more than one equally likely corrections, and also if any correction yields a well-formed ambiguous sentence.

Several natural language systems do not treat ambiguity (PLANES [Tennant 80], CO-OP [Kaplan 79], LUNAR [Woods 73] and SHRDLU [Winograd 71]), claiming that at most one interpretation was actually intended. These systems generally assume that the first good interpretation found is the one desired. Other systems have used ambiguity in the way the ASK System does for similar reasons, but postpone semantic processing until after disambiguation by interacting with the user (INTELLECT (ROBOT) [Harris 77], LIFER [Hendrix 78], RENDEZVOUS [Codd 78] and TQA (REQUEST) [Damerau 81]). Because of the overall system quickness, the ASK System can perform semantic processing before querying the user for a particular choice. This has the advantage of using the semantic information for disambiguation, and the interpretations that will not work (or would give semantic diagnostics) are not presented to the user unless those are the only interpretations available. However, no semantic processing takes place that would change either the language or

the underlying knowledge base if the input is ambiguous. How the user can make unambiguous input will be evident from the messages given for disambiguation.

The ASK System has used ambiguity for several reasons. First it is very natural to have one word with several meanings within a given setting. For example, the word "multiply" has several meanings even within several mathematical settings, depending on the type (integer, matrix) of operands that are being "multiplied". Both of these definitions would be expected in a math package, and the system would be expected to know which applies. Thus "multiply" is often disambiguated in context by grammar rules or semantic requirements, but remains ambiguous for the contexts that do not disambiguate it, such as "What is the definition of multiply?". Ambiguity is important secondly, because natural language is ambiguous (consider the example at the beginning of the chapter.) Thirdly, ambiguity handling capabilities are very helpful in keeping the various corrections on an equal footing. Because the ASK System uses ambiguity, messages are needed to explain the different interpretations made by the system. Furthermore, even within systems that do not allow ambiguity or try to find the interpretation intended, diagnostics are needed to explain input that is either rejected because of ambiguity, or would normally have multiple interpretations. Diagnosis is necessary whenever there is a chance of misunderstanding what the system has done.

5.2 Types

Six types of ambiguities may occur in the ASK System environment and these are the ones that must be diagnosed. The six types are structural, lexical, definitional, semantic,

anaphoral and correctional. A structural ambiguity comes from input that gives parsing charts that differ in their structure. Lexical ambiguities have similar structures but have elements of that structure differing in their part of speech or features. The structure is also the same for definitionally ambiguous input, but a defined word of the input has multiple definitions. Input that passes through all syntactic processing unambiguously but has several interpretations semantically is semantically ambiguous. If the input has a type of reference and there is more than one equally appropriate referent then the input is anaphorally ambiguous. Input is considered to be a correctional ambiguity if more than one correction has been successfully applied to it within the current level of correction.

5.3 Treatment of Individual Types

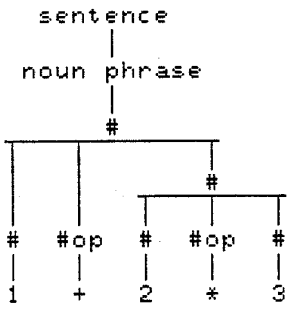
All ambiguous interpretations of a given input are considered equally. Diagnostics are prepared by the disambiguation routines to distinguish between these interpretations. Each type of ambiguity has a particular type of disambiguating diagnostic, which will be described here along with the method of its generation. How the different types interact within the overall framework of the disambiguating procedures will be discussed after the individual types are presented.

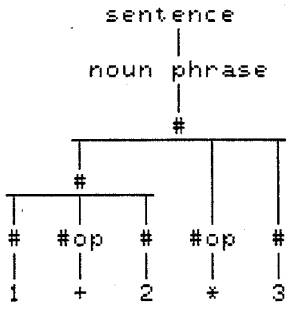
5.3.1 Structural Ambiguities

Parsing charts that differ in structure will be generated by sets of rules that do not give a clear precedence to their order of application. This may happen in natural language with

combinations of more than one relative clause or prepositional phrase. A simple example to illustrate differing parsing chart structure is integer four-function arithmetic without precedence of the operations. In this environment, 1+2*3 could be interpreted as either 1+(2*3) or (1+2)*3. Although there is an agreed upon precedence of these operations (that is included in the rules of the ASK System) there are other operations in both mathematics and English that have no well defined precedence.

In the arithmetic example, parentheses were used to distinguish between the two interpretations. This is the method that will be used to disambiguate structural ambiguities. Continuing this example, the following figure shows how the structure of the parsing chart can be captured by placing parentheses in the input. Here # indicates an integer and #op is an integer operator. Parentheses are added to the input string at each place where a rule application has occurred. The disambiguating message is then made by eliminating the parentheses that reflect the same structure in each interpretation.





(((1)(+)((2)(*)(3))))

1+(2*3)

((((1)(+)(2))(*)(3)))

(1+2)*3

Once the disambiguating messages are generated, the output can be formed. The resulting interaction with the user would look like:

```
>1+2*3
 There are 2 answers:
 1) 1+(2*3)
    7
 2) (1+2)*3
    9
```

For an example using English, a variant of the exchange at the opening of the chapter has this kind of ambiguity:

```
>Who is the man with a small cat named Lion?
 There are 2 answers:
 1) who is the man (with a small cat named Lion)?
    George Peabody
 2) who is the man (with a small cat) named Lion?
    Alfred Lion
```

The disambiguating messages are formed as in the previous example, using the differing structure of the parsing charts to provide parentheses to show how the grammar rules were applied. Here the parentheses indicate whether it is the man or the small cat that is named Lion.

It should be mentioned here that the knowledge base and semantic rules will often eliminate some interpretations. If there was no "small cat named Lion" in the preceding example, but there was a "man (with a small cat) named Lion" then the first interpretation would give the semantic diagnostic: "There is no small cat named Lion." This interpretation would then be suppressed by the outputter precedure, and disambiguating messages would be unnecessary and hence suppressed also. By these means the user is left unencumbered by interpretations that are semantically undesirable. This is how the "I rode down the street in a car" example of [Winograd 71, p. 89] is treated.

5.3.2 Lexical Ambiguities

Lexical ambiguities will often give rise to structural ambiguities, but occasionally will not. In this case, the parsing chart is searched for lexical ambiguities. Once tested for structural ambiguities, the structure of the various interpretations of the input may be assumed to be the same. To find the difference between lexical ambiguities then, the parts of speech and features are checked to see if they are all in agreement. Any differences found become the basis for the disambiguating messages. These messages just give the word in question along with its part of speech. This assumes some linguistic capability on the part of the user, but fortunately it is rare that lexical ambiguities ever yield more than one good semantic interpretation. It is not assumed that the user knows anything about features, thus such ambiguities may go undiagnosed.

Messages describing lexical ambiguities have the following form: for the input of "ships?":

```
"ships" has part of speech "noun-phrase"
"ships" has part of speech "verb-phrase"
```

Since only one of these interpretations will go through semantic processing correctly, the messages turn out not to be needed for this example.

5.3.3 Definitional Ambiguities

When one word is defined more than once potential definitional ambiguities come into the system. This happens particularly with abbreviations, function names (which are often really abbreviations) and with words that have a special meaning within the ASK System environment such as object or class.

Differences between definitional ambiguities are found the same way as lexical ambiguities. Parsing charts that are not structurally ambiguous are searched for distinct definitions. These definitions then give the message distinguishing the interpretations of the input:

```
>Who is the spouse of Smith?
 There are 2 answers:
 1) who is the spouse of Smith?
    "Smith" defined as "Bill Smith"
    Wilma Smith
 2) who is the spouse of Smith?
    "Smith" defined as "Alfred Smith"
    Freda Smith
```

The string that was retained from the original definition provides the meaning of the defined word. It is expected that the form of this definition as input by some user provides sufficient diagnosis of the nature of the ambiguity in an understandable manner. If an input is ambiguous because of underlying ambiguities, for instance if Smitty was defined as Smith in the preceding example and the question was "Who is the spouse of Smitty?", then the underlying definition will be in the parsing chart which will again provide the disambiguating information, but the intervening definition will also be mentioned.

5.3.4 Semantic Ambiguities

Semantic ambiguities are the cases where there are multiple interpretations because of something in a semantic routine executed after syntax processing (preprocessor, parser and whole arcs). An example of this is the intervening attribute procedure, where an attribute relating two noun phrases that are not directly related must be found. The disambiguating messages for any ambiguity arising within a semantic routine

after syntax time is the responsibility of that semantic procedure. The way this works with the intervening attribute procedure is illustrated with the following example, where the message comes from the attribute which was found.

```
>What are Boston ships?
 There are 2 answers:
 1)what are Boston (home port) ships?
   Maru
   Alamo
 2)what are Boston (location) ships?
   Oceana
```

5.3.5 Anaphoral Ambiguities

When elliptical fragments or input with pronouns can be resolved into complete sentences without any references in more than one way, then the input is anaphorally ambiguous. There are several syntactic, semantic and contextual considerations that limit what makes for a good reference resolution. These eliminate most of the possibilities. But occasionally there will remain more than one good result. Since diagnostics are needed to explain exactly what substitution was made, anaphoral ambiguities are distinguished by these same messages. The generation of these messages is accomplished by including whatever was missing from the input in the echo of what was evaluated. An example is given here to show how the interpretations are disambiguated.

```
>Who are George Peabody and Alfred Smith?
 George Peabody
 Alfred Smith
>Who is he?
 There are 2 answers:
 1)who is George Peabody?
   George Peabody
 2)who is Alfred Smith?
   Alfred Smith
```

The full discussion of how the messages are generated depends on how the references are resolved, which is covered in the chapter on anaphora.

5.3.6 Correctional Ambiguities

Since corrections need diagnosis even in the absence of other corrections, these diagnostic messages serve the function of disambiguating multiple corrections. Each valid correction of the input is echoed before its corresponding answer.

```
>List emales.
 The following word is not in the vocabulary: emales
 Corrections:
 There are 2 answers:
 1>List females.
   Bertha Peabody
   Wilma Smith
   Freda Smith
 2>List males.
   Bill Smith
   George Peabody
   Alfred Smith
```

These messages are obtained from the literal strings in the parsing chart located with the syntactic information about the sentences that the individual corrections produce. This example of an ambiguous spelling correction also shows the vocabulary diagnostic. Any such syntactic diagnostic will be given before echoing the corrections made to the input.

5.4 Putting Them All Together

An input may be ambiguous in more than just one of these six ways. In this case, several messages will be needed to communicate to the user the different interpretations of the input made by the system. The disambiguating routines are structured so as to first give the messages required by corrections or anaphor resolutions, and afterwards give further messages as necessary to fully capture the nature of the

interpretation. Messages for semantic ambiguities are generated independently within the appropriate semantic routine, and the other disambiguating procedures have been written knowing that such further message creation may take place after their execution.

Correctional disambiguation comes first. If the input has been corrected and is ambiguous, then the literal strings explaining the corrections are gathered for output. Duplicate messages indicate the need for further disambiguation. Next there is the resolution of references, which happens first if no corrections have been made. Any anaphoric constructions present will generate an explanatory message, diagnosing any anaphoral ambiguities in an interpretation. Remaining undistinguished ambiguities have their parsing charts searched for potential structural ambiguities. After this, ambiguities with similar structures are examined for lexical or definitional ambiguities. The following example shows several of these messages working together:

```
>Who is eh?
 The following word is not in the vocabulary: eh
 Corrections:
 There are 4 answers:
 1)who is he?
   who is the man (with a small cat named Lion)?
   "Lion" defined as "Siamese Lion"
   George Peabody
 2)who is he?
   who is the man (with a small cat named Lion)?
   "Lion" defined as "Persian Lion"
   Ned Jones
 3)who is he?
   who is the man (with a small cat) named Lion?
   "Lion" defined as "Alfred Lion"
   Alfred Lion
 4)who is he?
   who is the man (with a small cat) named Lion?
   "Lion" defined as "Bill Lion"
   Bill Lion
```

Fortunately, ambiguity in such combinations very rarely happens. Still, the capabilities were deemed necessary. It is not unreasonable to expect ambiguities to appear as a language is extended, but they need to be diagnosed so that the user may curtail the language or choose definitions more wisely if too many ambiguities start to occur.

Certainly there is something wrong with input that results in a large number of interpretations. If there are more than thirty interpretations, the disambiguation routines proceed no further than to attempt to diagnose the first thirty of these. The ambiguous inputs found in the protocol analysis never had more than two interpretations. Thus there should never be more than the number that has been chosen. If there ever is, the input causing this would make for interesting study itself, if the language had not been extended just for this purpose. But a diagnostic explaining all of the thirty interpretations does not seem to agree with the intuitive notion of habitability. This is a case where the language has become unhabitable, rather than the system.

This chapter will be closed with one final example. This example shows that while sentences that are in many ways ambiguous are of highly dubious value to interactive natural language systems, they may be quite entertaining.

> "I wondered what the neighbors would think about a man chasing a cat with a broom in his pajamas." --Jack Smith [Smith 81]

This also shows that, as the author points out, "some ideas simply can't be expressed in perfect syntax", and "grace and euphony may be more desirable than logical order." It is

interesting to consider how many ideas (interpretations: syntactic, semantic and anaphoric) this piece of syntax does indeed bring to mind.

Chapter 6: Anaphora

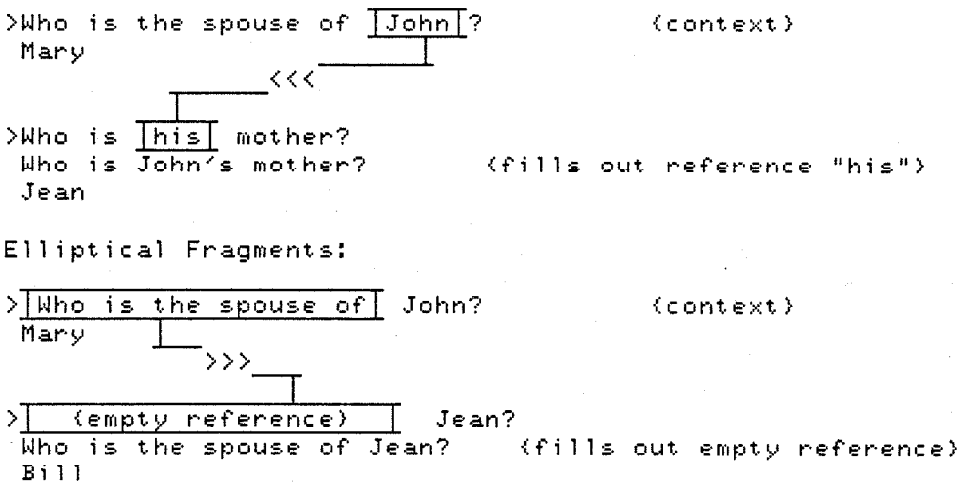
6.1 Introduction

> "Anaphora is the device of making in discourse an abbreviated reference to some entity (or entities) in the expectation that the perceiver of the discourse will be able to disabbreviate the reference and thereby determine the identity of the entity."
> [Hirst 81, p. 4]

In light of this definition of anaphora, the major task of anaphora resolution mechanisms is to find the identity of the entity to which abbreviated references were made in a particular context. After determining such a reference's identity, processing must use this identity and any further information added in the means of abbreviation to perform a complete execution of the input. It is desirable to be able to handle as many of these natural types of abbreviated reference as possible to improve the habitability of natural language systems. This is because these abbreviations are made for simplifying the form of the things that participants in a conversation say to each other. General habitability is improved by treating the most common types, and specific applications may need their own extra methods.

User input requiring anaphora resolution has been described earlier in this thesis as well-formed, incomplete input. The incompleteness is semantic, and comes from the abbreviated references. To form complete input, replacing the abbreviated references with complete references is necessary. These mechanisms are thus replacements in terms of modification of the original input, except in the ellipsis case. Elliptical fragments are completed by replacing the empty reference, which is an insertion into the user's original input.

Pronouns:



Elliptical Fragments:

Filling out incomplete well-formed input was described before as the "nearest" change that can be made to input. This chapter will further refine this idea by giving a (partial) order to a list of potential referents. The distance from the original input is then greater for potential referents found farther down the list. The ordering is partial to account for potential ambiguity in reference. Attempts at completing input will of course be made using closer changes first.

Questions pertaining to the treatment of anaphora in natural language systems have been the sole topic of several other theses, discussed below. Thus neither a thorough discussion nor treatment of all of these issues can be made in one chapter of this thesis, if at all. However, a subset of the types of anaphora, selected on the basis of their prevalence within the protocols, will be considered. These are the types that are most necessary for habitability within the general direction of application areas for ASK System languages. How anaphora resolution takes place within the larger system is also a primary consideration. The treatment of the chosen types

serves as a basis for the processing of other types, and some ideas for the processing of the other important anaphors will be presented. The method of retaining information for future reference is applicable to most of the other types, which then need only recognize that a particular type of reference has been made.

6.2 Types of Anaphora Considered

The primary anaphora forms considered are referential pronouns and elliptical fragments. The referential pronoun category consists of several subdivisions. This includes pronouns that reference a noun phrase within the same sentence or in a preceeding sentence. Pronouns may occur before the referent noun phrase in a sentence. Pronouns may reference a variety of noun phrases, such as the data structure provides, including temporal and locative references. Anything with the part of speech noun phrase is a potential referent for a pronoun.

Special attention is given to noun phrase elliptical fragments, particularly terse questions. Other terse fragments encountered will do the expected thing, such as added information, terse information, corrections and echoes. These will give a response even though a response is not really necessary in these cases. Terse responses are taken care of by each routine of the system that asks a question.

Elliptical fragments that are not noun phrases are also handled, but with somewhat less sophistication (a simple part of speech matching). The pronouns "I", "you" and "we", and their various forms ("your", "mine", "our" etc.), are processed correctly within the ASK System, but are not currently part of

anaphora processing. This is because they are clearly defined noun phrases within the system's interactive setting, since there are only two participants in the dialogue. "I" will always refer to the current user, "you" refers to the system and "we" refers to the combination. These pronouns would not be as clearly defined in a setting with more individuals involved. As the system develops to a multi-user, multi-computer environment allowing interaction among all parties, the treatment of these formerly well defined pronouns must be expanded.

Other types of anaphora not treated will be discussed near the end of this chapter, along with the prospects for processing them within the described setting.

6.3 The Theoretical Approach

The method of retaining information about the possible referents is the principal portion of the analysis and design relating to anaphora. The necessity of discourse orientation, retaining selected parts of the context of interaction and its major themes, is developed in [Hirst 81]. Hirst points out that a simple noun phrase recency list is insufficient for anaphor resolution by using examples to demonstrate this point. This result is confirmed empirically by the use of pronouns in the protocols, even in the human-to-computer mode. In this mode, there was only one instance of using a pronoun with its referent in the same sentence:

>LIST EACH CARGO AND ITS DIMENSIONS ON THE WELL DECK OF THE ALAMO.

But there were several examples of using pronouns to repeatedly refer to one noun phrase, which would have been eliminated before the later references from a recency list of noun phrases:

```
>IS THERE A DECK WHOSE PRIMARY USE IS AMMUNITION AND
 WHOSE LENGTH IS 396?
 YES
>LIST IT.
 MAGAZINE 3-37-1-M
 MAGAZINE 3-37-1-M
>LIST THEIR SHIPS.
 USS PEORIA
 USS SAN BERNARDINO
>LIST THEIR LOCATIONS.
 THERE ARE NONE.
>LIST THEIR TYPES.
 THERE ARE NONE.
```

By far the most common usage was to refer to a noun phrase in the preceding sentence. These conclusions were also true of the other experimental modes, although the other modes occasionally had more obscure reference types such as references to concepts related to the discussion but not previously explicitly mentioned. These other types of reference further point out the inadequacy of the recency list approach. How a noun phrase list may be maintained, updated and supplemented to reflect the themes of the discourse is thus what is most lacking from the noun phrase recency list approach.

How can the proper concepts for potential reference be selected? This has been the topic of a few recent important theses in computational linguistics. These approaches will be contrasted here with the chosen (theoretical) method that the algorithm to be presented is based on. For a more complete survey of other work in this area, the reader is referred to [Hirst 81].

Two words that have special meanings within the literature of anaphora will be defined here. First is focus, which is the set of concepts available for reference in a given context. The second is theme, which is what a discussion is about. This is the usual idea of theme, and the focus is that part of the

theme that is currently dominating the interaction, the part being focused on. These words will not be any more explicitly defined, as they have different meanings depending on which paper you read. Within the computational setting, the focus can be thought of as the list of concepts available for reference. A particular theory of anaphora treatment is determined mostly by how it maintains such a list.

The approach taken here is a combination and extension of the methods presented in [Hajicova 82] and [Roach 80]. The latter describes how pronouns may be matched to their referents by primarily syntactic conditions. The procedure used for this process, known as chaining, is essentially the same as in Roach's thesis, with some changes made to the data structures. Other programs in the anaphora resolution algorithm (particularly interpret) have been influenced by his work. The vocabulary describing nodes, relations between nodes and the syntactic conditions on chaining are all derived from his thesis.

The former paper [Hajicova 82] is where the ideas for focus maintenance come from. The notion of the salience of a concept is defined here along with rules for computing and updating values representing the salience. This will be called the salience weight in the sequel to try to minimize the confusion arising from using increasing salience weights to correspond to concepts with lower salience. The rules have been adapted to the ASK System setting by making explicit the notion of "immediately associated", and some rules have been added to curtail the size of the focus.

Thus the pronoun portion of the algorithm to be described is rooted primarily in these two papers, along with a few contributions from the ones mentioned below. The contributions of Roach have been the syntactic conditions needed mainly for resolution of pronouns referring to noun phrases within the same sentence. The paper by Hajicova and Vbrova gives most of the rules for maintaining the ordering on the list of potential referent noun phrases, and hence forms a basis for the discourse orientation of the anaphora resolution mechanism.

The current thesis puts all these things together with some refinements (such as the treatment of underlying noun phrases and the addition of several salience weight rules), within a setting considering an overall treatment of modification of user input and diagnosis. Methods for the pronoun parser and semantics have been developed, allowing for ambiguous interpretations that may arise either before or during processing. The resulting algorithm for pronoun resolution is then applied to many other types of anaphora, particularly elliptical fragments (including the terse fragments).

6.4 Comparison With Other Methods

All the approaches considered here are modern in the sense of [Hirst 81], as is the chosen algorithm. Traditional anaphora resolution, consisting of a noun phrase recency list with heuristics of varying degrees of sophistication, have already been shown to be lacking.

The first method considered is that of Grosz ([Grosz 77a, 77b, 78]). Focus is determined here by the results of studying protocols simulating a task oriented discussion. These protocols show what concepts are available for reference within given

sub-dialogues corresponding to sub-tasks of the problem. Focus management is structured on the basis of the analysis of these sub-tasks. This kind of design is useful for specific habitability, and confirms the idea that protocol analysis is the best way to prepare for and evaluate the habitability of a system. But it is clear that this approach is limited to the tasks that have been studied. Studies such as this may provide valuable supplement to the focus (e.g., concepts not found in the dialogue itself), and should be kept in mind by the language designer.

A generalization of Grosz's work is that of Sidner ([Bullwinkle 77a, 77b], [Sidner 78a, 78b, 79]), which uses situational knowledge already incorporated into the system to manipulate the focus. This treatment of focus relies heavily on the form of knowledge representation. In the earlier papers, frames served as both the focus and the knowledge representation. The concepts available for reference were those available in the active frame. Later [Sidner 79] an association network is proposed instead of frames. These methods point out the ties between focus and knowledge representation. They also show that focus shifting and initialization are at least as troublesome as choosing which parts of a knowledge representation to activate in a changing discussion. Thus extra care is needed when this kind of shift takes place. This corresponds to changing application languages in the ASK System. Since this is a more radical change than shifting frames, a change clears the set of potential references and sets up a new one, adding anything necessary for that language.

Sidner has continued study in this area, described in [Sidner 81]. This gives a set of pronoun interpretation rules that cover a wide range of pronoun usage. Many of the requirements that these rules specify are met by the salience management rules. One primary difference is the use of separate foci for different cases in Sidner's theory. Although available for use within the theory presented here, resolution methods presently incorporated have not needed overt case information.

Another current method for focus management developed by Webber ([Nash-Webber 76, 77], [Webber 78a, 78b]) uses previous sentences in a text to generate the set of potential referents rather than prior knowledge as in the first two approaches. This method uses logical forms of input sentences to consider in detail some of the subtle problems of specific entities for reference. These specific entities are then available for future reference, but unfortunately are not for the current sentence. Several formal results about the use of anaphora with quantifiers have been contributed by these studies.

In [Hayes 81a], an anaphora handling approach is presented which also considers reference treatment within the larger setting. The aim has been to provide users with an "equivalent functionality" rather than complete modeling of human performance, whenever this performance was "impractical to reproduce." It is necessary then to provide teaching and explanation facilities for the user about these new methods for reference.

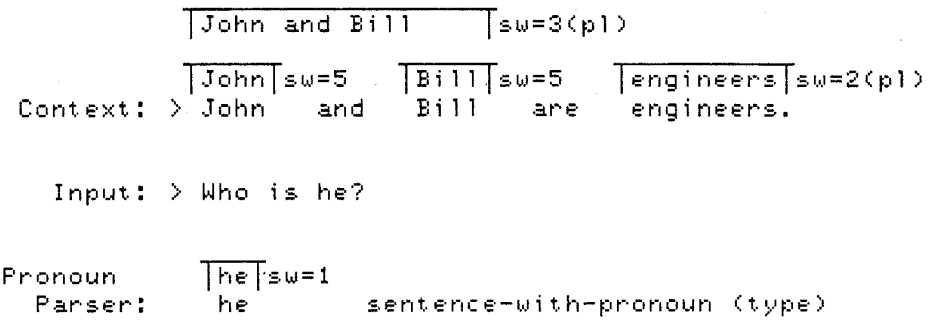
The idea of the activatedness of a concept is described in [Kantor 77], where this is used to explain the difficulty in resolving some pronouns. This is quite similar to the salience of a concept. Salience has its roots in communicative dynamism (see [Sgall 77]), which Kantor distinguishes from activatedness. Kantor also gives some ideas of primary and secondary competence in using and resolving pronouns, which may be used to gauge a system's capabilities.

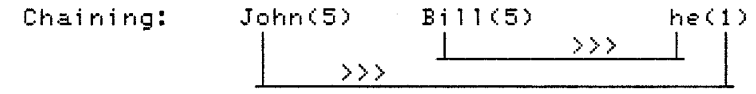
6.5 The Practical Approach

The approach taken here is more of a practical one than most of the other work that has been considered (except [Hayes 81a]). This is because while a complete system modeling anaphoric reference in natural language would be preferred, there are other things that get in the way even if such a goal were achieved. One of these is that people do not necessarily behave according to theory. A system might resolve a reference in the most reasonable way, but this does not guarantee that this will agree with the user's intent (thus the necessity of echoes). Furthermore, actual errors may be made in the use of anaphoric reference, violating either the particular theory or normal human performance. Hence anaphora should be considered as it fits into an overall setting of modifying user input. For a system that is to be used, too much time in both design and computation can be easily (but not habitably) spent attempting to yield a "perfect" (in terms of competence) system.

6.6 Pronouns: Overview of the Algorithm

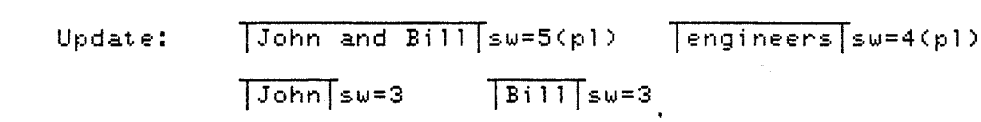
Description of the anaphora mechanisms will begin with the pronoun handling capabilities, as these subsequently serve as routines available for use in treating other reference types. These might also be called the noun phrase anaphora routines, as noun phrases represent the concepts available for abbreviated references that are manipulated by these routines. There are five major portions of the pronoun resolution procedure: the pronoun parser, chaining, interpret, semantics and update. The pronoun parser finds and saves noun phrases. It also determines relationships between noun phrases and assigns initial salience weights to them. The chaining program links noun phrases to pronouns that may pronominalize them. The interpret algorithm finds all legitimate (non-conflicting) combinations of these links to form interpretations. The semantics procedure modifies and evaluates user input to see if any of the interpretations make sense. The update routine updates the salience weights of the retained noun phrases and then determines which noun phrases to save. The individual parts of the algorithm are described in detail in the next sections. Their functions are illustrated in the following diagram (sw indicates the salience weights and pl means plural).





Interpret:
1) John--->>>---he(6)
2) Bill--->>>---he(6)

Semantics:
1) Who is John? John
2) Who is Bill? Bill



6.6.1 Main Dispatching Procedure

These procedures are coordinated by a calling program which is principally for ambiguity management. Input may be ambiguous either before or after pronoun resolution, or both before and after. This routine also controls access to the others, as only the pronoun parser and the update programs are necessary if there are no anaphors in the input. Also, there are a few special commands that need exceptional treatment, such as those for entering and exiting from different languages, that are taken care of by the administrating procedure.

The main calling program works as follows. It is called by the whole arcs procedure which gives to it the list of the interpretations found (thus far) of the user's input. For each of these ambiguities, the pronoun parser, chaining, interpret and semantics are called. If there is no pronoun or elliptical noun phrase in the input, only the pronoun parser is called. After each of the ambiguities have been processed this way, the update procedure is invoked with a list of the good interpretations of any anaphors that have been resolved. Updating is performed differently during language changes, which

normally clears the list of potential referents but may force new ones onto the list while entering particular languages. Some final editing of the list of ambiguous interpretations is done before returning control to the whole arcs program.

6.6.2 Pronoun Parser

There are four primary responsibilities of the pronoun parser. It must decide on the general type of anaphora, if any. The nodes of the parsing chart that are important to the resolution procedure must be found and the relationships between them determined. Lastly, the initial values for the salience weight of the noun phrases in the input need to be computed. All of these things take place together, but can be thought of as separate tasks of the program.

The pronoun parser distinguishes between four general types of input: sentence without pronoun, sentence with pronoun, fragment without pronoun and fragment with pronoun. These types influence the processing by the other routines.

6.6.2.1 Nodes

Certain elements of the parsing chart are of particular interest to the pronoun routines and will be called nodes. These are determined by the parts of speech (and in some cases the features) in the parsing chart. There is also a node type used only by the algorithm itself in the chaining process.

Nodes are used to find the relationship between noun phrases in preparation for the testing of syntactic conditions by the chaining routine. The node types are:

S-nodes — Sentence or underlying sentence nodes: includes such parts of speech as "sentence", "verb-phrase", "relative-clause".

C-nodes Conjunctions of other nodes: includes only the parts of speech "noun-phrase" and "verb-phrase" with the conjunction feature set.

N-nodes Noun phrase nodes: anything with part of speech "noun-phrase".

E-nodes Nodes used for chaining.

Notice that some arcs may be more than one node. There will be an N-node made for each noun phrase in the parsing chart, which will contain all the information needed for pronoun resolution. Even without any anaphors in the input the N-nodes are needed for adding to the potential referent list.

6.6.2.2 Relations Between Nodes

The relations between N-nodes for testing syntactic conditions are given here. These relations need only be calculated for N-node pairs with at least one N-node corresponding to a pronoun. The values for N-nodes not from the current input in relation to any in the input are given as default values (consider node A to be any member of the potential referent list and node B to be an N-node of the current input).

1) Dominate: (False)
   A node A dominates another node B if:
      B is a contributing (underlying) node of A.

Each of the following relations between nodes A and B require that neither A nor B dominates the other:

2) Precede: (True)
   A node A precedes another node B if:
      A occurs before B in the input.

3) Command: (False)
   A node A commands another node B if:
      the S-node that most immediately dominates A also dominates B.

4) Separate: (True)
A node A is separate from another node B if:
the lowest node in the chart dominating both
A and B is a C-node.

5) Same Simplex: (False)
A node A is in the same simplex as another node B if:
the lowest node in the chart dominating A is
the lowest node in the chart dominating B.

These relations can be thought of as positions within a parsing graph or tree representation of the parsing chart. Dominate is then the same as ancestor node, and same simplex is having the same immediate parent node. Separate is when the lowest common ancestor of both nodes is a C-node. Precede means that one node occurs before the other in an inorder transversal, and command means that the first parent S-node is also an ancestor of the other node.

6.6.2.3 Salience Weight Assignment

The pronoun parser computes the initial values for the salience weight of noun phrases found in the input. This is done on the basis of several salience weight rules. These rules are listed here along with the number of the corresponding rule in [Hajicova 82]. Where no number is indicated, this is a new rule. The names of the rules have also been added for future reference. The actual weights assigned are given, but occur as parameters in the program so that they may be calibrated for specific applications. The names following these weights are the parameter names. First the topic and focus of a sentence are defined.

Sentence Topic/Focus

The Topic of a sentence is the surface subject.
The Focus of a sentence is the surface predicate.

Initial Salience Weight Assignment

Focus Rule (2): A noun phrase in the focus position will receive a salience weight of 0 (=initial focus salience).

Topic Rule (3): A noun phrase in the topic position will receive a salience weight of 1 (=initial topic salience).

Other Non-Dominated NP Rule: Any other noun phrases not dominated by another noun phrase (e.g. indirect object) will receive the initial topic salience also.

Other Dominated NP Rule (cf. 4): Any noun phrase that is dominated by another noun phrase will receive a salience weight equal to the salience weight of the dominating noun phrase plus 2 (=salience increment).

The topic and focus rules are the same as in [Hajicova 82]. The other non-dominated NP rule is for noun phrases that are neither in the focus or topic position, nor underlying one of these positions. This is necessary for the ASK System setting because the formal grammar representation is somewhat different.

The topic rule also applies to noun phrases that occur by themselves or in fragments, and thus includes the functions of rule (5) of [Hajicova 82]. This sets salience weights for such beginning fragments as "as for x", and "concerning x". Terse information fragments also commonly occur for the purpose of setting the salience weight of a noun phrase, and the topic rule given here covers this kind of action.

The other dominated NP rule is for underlying noun phrases. This may be compared with rule (4) of [Hajicova 82], which says that if an object receives a salience weight of n, any object "immediately associated" with it obtains a salience weight of n+2. Underlying noun phrases are considered to be one kind of immediate association, but by no means cover all associated objects. Other rules may be hypothesized to cover the association necessary for referents not found in the input, such as the following:

> Immediately Associated NP Rule (cf. 4): Any noun phrase that is one link away in the knowledge base from a noun phrase in the input receives a salience weight of that noun phrase's salience weight plus 2 (=salience increment).
>
> Associated NP Rule (cf. 4): Any noun phrase related by some link in the knowledge base to a noun phrase with an assigned salience weight will receive a salience weight of that noun phrase's salience weight plus 2 (=salience increment) until such a point that assigned saliences would be greater than 7 (=maximum salience).

Neither of these rules has been implemented, but show that rules can be devised to make more explicit to a given knowledge base setting the notion of "immediately associated" objects. Some association is covered with the other dominated NP rule, and in a setting where this type of reference occurs more frequently the hypothesized rules may be added. It was decided that these were unnecessary for most applications, and within the structure could be added to the system fairly easily later if deemed useful in some applications languages.

Application of each salience weight rule is demonstrated in the following example, where again sw is the salience weight.

```
| John and Bill |sw=1

|John|sw=3 |Bill|sw=3  |flowers|sw=0  |Mary|sw=1

> John  and  Bill  gave  flowers  to  Mary.
```

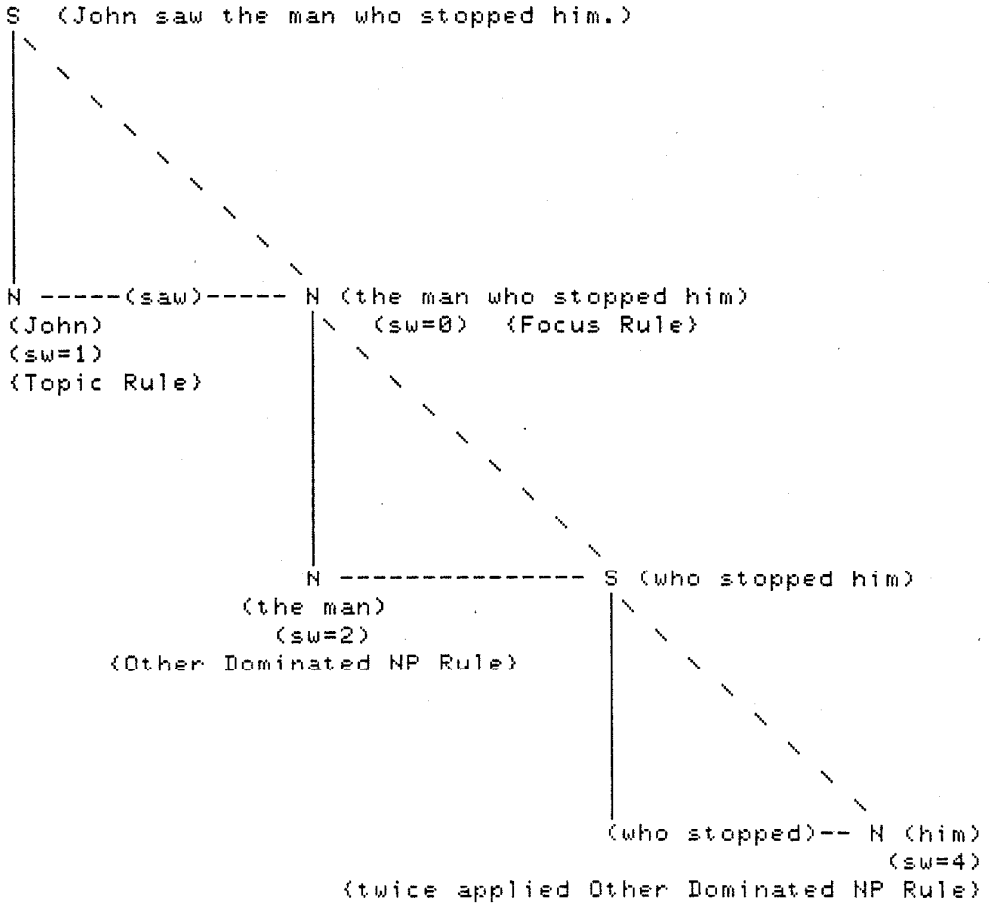
Topic Rule: "John and Bill" has sw=1
Focus Rule: "flowers" has sw=0
Other Non-Dominated NP Rule: "Mary" has sw=1
Other Dominated NP Rule: "John" and "Bill" have sw=3

It should be pointed out that ambiguous interpretations of input form ambiguities in the list of noun phrases by yielding equivalent salience weights. Equal salience weights are also often assigned within the structure of the other dominated NP rule. This is to achieve the appropriate anaphoral ambiguities.

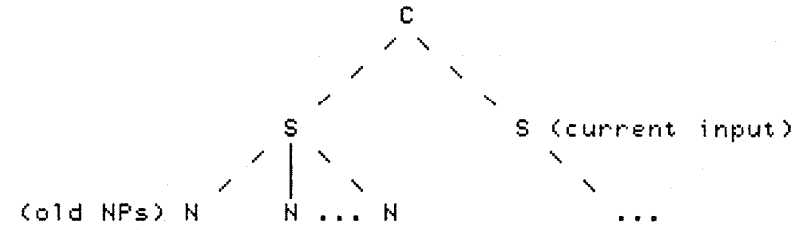
All incorrect ambiguous interpretations are expected to be deleted when semantic processing takes place.

6.6.2.4 Example of Nodes and Salience Weights

The example given below shows nodes and initial salience weight assignments for a sentence. How this information is used is discussed in the parts describing the chaining and semantics programs. Salience weights (sw) are indicated along with the rules giving the individual assignments. The relations between the nodes can be seen from this representation. Only the dominate relation is of interest here. Nodes are dominated by nodes above them, but not beside them or beside nodes above them.



The following diagram shows how the noun phrases saved from previous input fit into this scheme (as nodes):



This form motivates the assignment of values to relations between nodes given above.

6.6.3 Chaining

Chaining is the process of linking noun phrases to pronouns that may refer back to them, according to syntactic conditions. This is done by creating an E-node attached below the appropriate N-node, and then linking the E-node to an E-node below the pronoun's N-node. Several E-nodes from the same noun phrase N-node may be made this way, and the ambiguous interpretations this represents are collected by the interpret routine.

6.6.3.1 Syntactic Conditions on Chaining

Three requirements must be satisfied for a pronoun to be subject to chaining to a noun phrase. These are tested by the chaining procedure, and the chaining E-node is generated when these conditions are met. They are listed here:

Precede and Command (or Separate) Rule
A pronoun P may be used to pronominalize a noun phrase NP unless both:
a) P precedes NP, and
b) P commands NP or P is separate from NP.

Feature Agreement Rule
A pronoun P may be used to pronominalize
a noun phrase NP if:
P agrees with NP on the following features, or
P is explicitly relaxable on disagreeing features:
a) animate,
b) gender, and
c) plural.

Reflexive-Non-Reflexive and Genitive Rule
A pronoun P may be used to pronominalize
a noun phrase NP if:
for reflexive P:
NP is in the same simplex and not genitive,
for non-reflexive P:
NP is outside the same simplex, or genitive.

These rules contain the basic syntactic conditions for pronoun resolution. The feature agreement rule is fairly loose, in that it might be explicitly overidden by other parts of the system. The other rules are mainly for specific types of pronominal usage. The precede and command rule tells when a pronoun may precede the noun phrase it refers to. The final rule is for the interpretation of reflexive pronouns. The use of these rules can be seen from these examples (adapted from [Roach 80]):

1)The man who called to it caught the barking dog.
2)John took June to dinner, but he isn't sure she liked it.
3)Mary's mother voted for herself.
4)Mary's mother voted for her.

The first rule applies to the first example by showing that although "it" precedes "the barking dog", the pronoun neither commands or is separate from the referent noun phrase. Thus this pronoun use is permissible. The second example is of gender and animate restrictions. The final two examples demonstrate the use of the third rule, where "herself" refers to "Mary's mother" and "her" refers to "Mary".

Notice that these chaining conditions also distinguish appropriate usages for pronouns referring to noun phrases in earlier input. (The precede and command rule is always satisfied for the noun phrases of an earlier input).

6.6.4 Interpret

The interpret program finds all legitimate sets of interpretations of pronouns in the input. The requirement for a valid interpretation is that all pronouns are resolved. This is determined by counting the number of N-nodes first, and comparing this number with the number of N-nodes interpreted by a combination of E-nodes. If these numbers are equal, then the E-node combination is a valid interpretation. A list of such interpretations is generated for use by the semantics routine.

6.6.5 Semantics

The semantics procedure of the anaphora resolution mechanism makes the changes in the input as required by the interpretations found. It then evaluates the modified input, and determines which interpretations were good for later modification of the potential referent list. Messages explaining the resolution of anaphors are generated for the good interpretations. If more than one interpretation (or ambiguity of the input before resolution) produces the same string for evaluation, it will not be resubmitted. The program maintains a list of attempted submissions for evaluation, to keep from repeating work and to treat ambiguity correctly.

6.6.5.1 Order of Evaluation

The (possibly several) interpretations produced by the chaining and interpret algorithms are evaluated in the order given by the total of the salience weights of the noun phrases in each interpretation. This is because those interpretations with lower salience weights are more likely to be the correct ones, according to the theory. An abbreviated reference, once passing syntactic conditions, is more likely to refer to a concept of higher salience. With this ordering, more evaluation attempts are necessary for the more obscure (less salient) types of reference. Ambiguities occur whenever more than one interpretation has the same salience weight. Once a semantically valid interpretation is found, only those with the same salience weight are tried, and no more. If an interpretation gives a (semantic) diagnostic, then the search continues, but it will be output if no good (non-diagnostic) results are found. In this way semantically invalid interpretations are eliminated.

6.6.5.2 Generation of Explanatory Messages

Whenever anaphors exist within a user's input, explanatory messages must be generated to make explicit what was evaluated, or if nothing reached completion semantically, a diagnostic is necessary. An echo serves to explain resolutions of abbreviated references.

If there was no semantically good result then the diagnostic message for the pronoun in question will be of the following form:

```
 (empty context)
>what is it?
 There are no good interpretations for the pronoun "it".
```

If there was a good result from semantics then an echo of the input sentence reflecting the change is output:

```
>what is the HP9836?            (context)
 HP9836
>what is it?
 what is the HP9836?
 HP9836
```

The diagnostic may be followed by interpretations giving semantic diagnostics:

```
>what is the area of a sheet of paper?    (context)
 93.5 square inches
>what is the volume of it?
 There are 2 answers:
 1)what is the volume of it?
   There are no good interpretations for the pronoun "it".
 2)what is the volume of a sheet of paper?
   There is no volume of a sheet of paper.
```

These messages are made from the strings contained in the interpretations.

6.6.6 Update

The update procedure performs the necessary functions for management of the discourse focus (the noun phrase potential referent list). First the focus will be described, and then the execution of rules for updating the focus will be discussed.

The discourse focus consists of a list of N-nodes, with an attached first E-node. Between each pair, the information about the noun phrase necessary for future reference is kept. This information is: the string for the noun phrase, the input string in which it was found and its location within that string, the salience weight and the relevant features. These features are yes/no bits for genetive, reflexive, animate, gender and neuter. There is also a feature to indicate the need for an apostrophe-s construction in the output message, which turns out to be slightly different from the genetive feature. Two

other features are kept for the noun phrase elliptical fragment resolution portion of the algorithm, the complex and no-ellipsis features. Their functions will be described later.

At the begining of the update routine, the list of noun phrases in the current input are added to the discourse focus. Duplicates are removed by keeping the most recent noun phrase and giving it the minimun of the salience weights. All pronoun N-nodes are deleted.

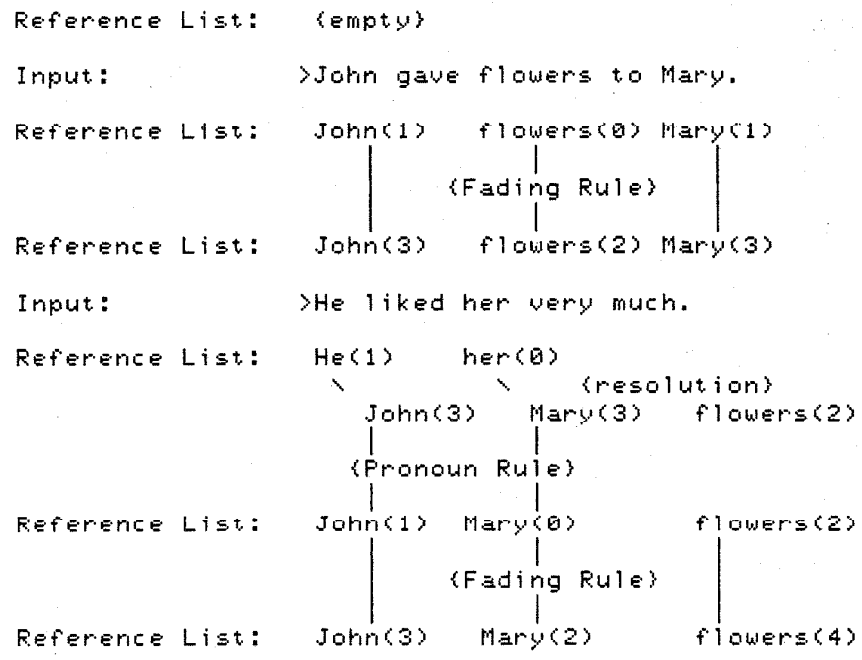
6.6.6.1 Update Rules

After the completion of processing by each of the previous anaphora programs on each ambiguity of the input, the update routine is executed. Interpretations that produced good output semantically are in a list of good interpretations. These are the update rules:

> Pronoun Rule (1): If a noun phrase occurs in a good interpretation, then the salience weight of the noun phrase is replaced by the salience weight of the pronoun referring to the noun phrase as the pronoun occurs in the input (if more than one, use the minimum.)
>
> Fading Rule (6): If a noun phrase does not occur in a good interpretation, then the salience weight of the noun phrase is increased by 2 (=salience increment.)
>
> Trimming Rule: Noun phrases that have a salience weight greater than 7 (=maximum salience) after the application of all the other rules are removed from the potential referent list.

The pronoun and fading rule are from [Hajicova 82], where the number corresponds to the rule number given in that paper. The pronoun rule is for keeping concepts referred to salient, even when that reference is abbreviated. The fading rule is to ease concepts out of focus by increasing the salience weight. The trimming rule is to prevent the potential referent list from growing too large, and to eliminate nodes that are no longer

considered to be salient. These rules are also parameterized for calibration to specific applications. The following example illustrates the application of the Pronoun and Fading Rules.



6.6.7 Some Final Remarks about Pronouns

The description of the pronoun resolution mechanism is now complete. A complete list of the pronouns treated and the features associated with each is given in appendix 3. Some of these pronouns may be lexically ambiguous, but in instances where they are interpreted as pronouns all are treated equally by the anaphora algorithm.

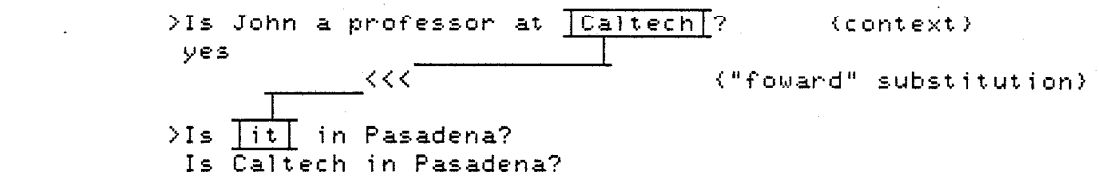
An example is given here showing how the salience, semantic, and syntactic considerations work together for the resolution of pronouns:

```
>What is the destination of the Oceana?
 Tokyo
>What is its country?
 What is the destination of the Oceana's country?
 Japan
>What is its speed?
 What is the Oceana's speed?
 22 knots
>Is that the speed of the Maru?
 Is the Oceana's speed the speed of the Maru?
 yes
```

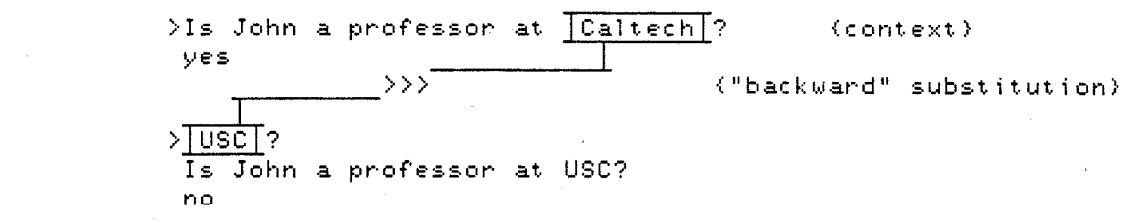
6.7 Elliptical Fragments: Noun Phrases

The next example highlights the similarities between pronouns and elliptical fragments that are noun phrases. The algorithm for processing elliptical noun phrase fragments is based on the observation that a solitary noun phrase can often be interpreted as a pronoun (to find a referent), and then substituted into the referent's location in the previous sentence.

Pronoun:



Elliptical Fragment: (Noun Phrase)



The treatment of elliptical fragments that are noun phrases is greatly facilitated by the pronoun algorithm and discourse focus. The discourse focus has all the relevant information

about previous noun phrases for future abbreviated reference. A solitary input noun phrase is processed by making it look like a pronoun to the rest of the procedures. Once referents have been found for this psuedo-pronoun, then this noun phrase can be substituted into the previous input at the location of its referent. The semantics routine knows that substitutions may occur either forward (in terms of the link, not the order of input) for pronouns or backward for isolated noun phrases. The rest of the description of the treatment of noun phrase elliptical fragments is just listing the other exceptions to this general rule of backwards substitution of these psuedo-pronouns.

6.7.1 Differences from Pronouns

Elliptical fragments are assumed to somehow fit into the previous input, but to change that input in a useful way by making a new request. Often this will mean only a slight change, such as moving from one member of a class to another, or changing a small portion of the syntactic features. Thus feature checks are not made when attempting to chain noun phrase elliptical fragments. Neither are substitutions attempted that would repeat the processing of a previous sentence.

The messages for elliptical fragments echo the completion of the input. If no resolutions are found for the psuedo-pronoun, then the noun phrase is evaluated just as it is input.

```
>who is spouse of John?
 Mary
>Mary?
 who is spouse of Mary?
 John
>Bill's height?
 5 feet 11 inches
```

This example shows noun phrase elliptical fragment resolution in context. It also shows how a change in feature can be expected in elliptical fragments (gender in this case), the resolution echo, and the evaluation of an isolated noun phrase with no elliptical fragment resolutions.

It could be claimed that there was a resolution of the elliptical fragment in the last input of the previous example. This demonstrates another important part of the noun phrase elliptical fragment processing. No substitutions are made that would give the same result as evaluation of the noun phrase by itself. This kind of elliptical resolution is implied by the way isolated noun phrases are evaluated by the system. The no-ellipsis feature of N-nodes in the discourse focus indicates whether or not the input containing the earlier noun phrase just evaluates that noun phrase or does something more.

When a solitary input noun phrase is composed of more than one noun phrase, it is considered to be complex. This distinction from simple noun phrases is necessary because there is some ambiguity when a complex noun phrase is input. The user may want elliptical resolution or evaluation of the noun phrase by itself. This ambiguity is made by the system whenever an isolated complex noun phrase is input. This example shows the ambiguity in context, with the resulting output.

```
>who is John?
 John
>spouse of John?
 Mary
>what is the height of the spouse of John?
 5 feet 5 inches
>spouse of Bill?
 There are 2 answers:
 1)spouse of Bill?
   Sue
 2)what is the height of the spouse of Bill?
   5 feet 4 inches
```

The complex feature on N-nodes in the potential referent list is a flag for complex noun phrases.

Another type of complex noun phrase is one containing a pronoun. An input noun phrase containing a pronoun is only evaluated as a noun phrase, with no elliptical resolution attempted. The pronoun is of course resolved. Thus isolated input noun phrases with pronouns are treated the same as sentences with pronouns:

```
>who is spouse of Mary?
 Bill
>his mother?
 Bill's mother?
 Martha
```

Salience weight management is essentially the same as with pronouns, except that now good interpretations contain noun phrases chained to noun phrases. Thus when the pronoun rule applies to a good interpretation, the N-node that was found for substituting the isolated noun phrase into receives the same salience as the elliptical fragment. The initial salience weight assignment to an elliptical noun phrase fragment is given by the topic rule as before.

6.8 Other Whole Phrases

Input that does not wholly parse into either of the parts of speech sentence or noun phrase but is covered completely by

a single arc of the parsing chart is considered to be elliptical. This kind of fragmentary input is less common than the noun phrase elliptical fragment types. The mechanism for resolving fragments other than noun phrases works for all parts of speech, but will not be called for sentence or noun phrase input. It serves as the final measure for treating elliptical forms for which there is no processing elsewhere. Thus as more theory is developed about these other elliptical fragments, other routines may update the system performance while any leftover forms will continue to be handled by this procedure.

The non-sentence whole-fragment program is invoked by the whole arcs procedure. It works by searching the parsing charts of previously input sentences for arcs with the same part of speech. Having the same part of speech is the only syntactic requirement for the analogue of (psuedo-) pronoun/noun phrase chaining. If more than one arc with the same part of speech is found in one of the previous sentences, then the different interpretations are considered to be equally likely ambiguities. Semantic evaluation is attempted on each interpretation, after substitution in the previous input. Salience of concepts is maintained by recency and by retaining only complete, well-formed sentences in the sentence list. Thus elliptical reference by other parts of speech keep these concepts salient by saving only the resolved input.

An example of the output messages, like those of elliptical noun phrase fragments is given here for a prepositional phrase:

```
>who is the mother of John?
 Martha
>of Mary?
 who is the mother of Mary?
 Alice
```

A case even more rare than the non-noun phrase elliptical fragments is the elliptical fragment that is not covered by a whole arc. Although it is fairly easy to make up an example, only one instance of this usage was found in the human-to-computer protocols (shown here as it should work):

```
>How many ships have a beam greater than 1000?
 19
>How many have a beam less than 1000?
 How many ships have a beam less than 1000?
 15
```

Selected cases like this are treated by additions to the grammar to form the whole arcs necessary for resolution by the whole arc non-sentence algorithm. This example works by using a grammar rule that adds quantifiers (how many) onto verb phrases in the correct way. This does not interfere with other processing except for the time required to use an extra grammar rule, which only affects the parser and is minimal.

6.9 Prospects for Anaphora Types not Treated

There is a great number of types of abbreviated reference that have not been treated by the anaphora mechanisms presented here. None of these has been considered to have a great enough frequency of use within the practical approach for general habitability to be necessary for implementation. To show the sturdiness of the system as implemented, the prospects for additions to treat some of these remaining types will be described. No claim is made that the sum of these proposals and the implemented algorithm constitutes a full treatment of anaphora or of the individual anaphora types. Rather, this discussion shows that many anaphora types can be found with intriguing properties of their own, and also serves to indicate

the kinds of anaphora that have been considered for use in improving habitability.

The other types to be mentioned fall into three categories: those handled by addition of special routines possibly invoked by features on the pronouns, parts of speech for which a pro-form/elliptical fragment algorithm may be generalized from the noun phrase one, and those requiring greater "semantic" knowledge possibly along with inference (which amounts to being a long way off). The names for the reference types given here are from [Hirst 81] (with the exception of set selection, which is from [Hayes 81a]).

6.9.1 Special Purpose Routines

The addition of several special purpose procedures would allow the anaphora algorithm to process correctly a few more types of abbreviated reference. A way to distinguish some of these from the usual referential pronoun is by the use of syntactic features. Three kinds of anaphora are considered here.

Set selection pronouns ([Hayes 81a] and [Thompson 78, pp.42-47]) are pronouns to refer to elements of a set that is currently salient. These include the pronouns "other", "another", "some" and "next", and are used as in "What are the other ships?" or "What are the others?". A flag on these pronouns would distinguish that they need special processing. This processing would take the form of incorporating the noun phrase following the pronoun into the chaining requirements for the type with a following noun phrase (the "other ships" type). Then a set subtraction will find the referent:

list the submarines.
list the other ships.

The type with only the set selection pronoun must apply the pronoun resolution mechanism twice, first to find the already selected portion of the set, and then the set itself:

list the ships.
list the submarines.
list the others.

(This second application may be accomplished by careful examination of the less salient interpretations.) In both examples, the referent is "ships not submarines".

Non-referential pronouns also need special treatment. This consists of usages like "It is fortunate that the food is good" or "It is one o'clock". Pronouns that can be used in a non-referential way would be lexically ambiguous, with the one marked as non-referential being ignored if an interpretation can be found. If no referent is found, an attempt is made to see if the pronoun is a place holding subject in a cleft sentence. This would produce "That the food is good is fortunate" in the first example, after suitable syntactic manipulation. If this fails, then the pronoun would be assumed to refer to the general environment. This is similar to the use of the pronouns "I" and "you", although it is a bit more esoteric to keep track of this information.

Some forms of strained anaphora can be handled by careful addition of some "immediately associated" salience weight assignment rules. An example of strained anaphora is taken from [Hirst 81]: "John became a guitarist because he thought that it was a beautiful instrument." Here the it refers to "guitar". This would be brought into the discourse focus by "immediately associated" rules applying to "guitarist".

6.9.2 Generalized Pro-Form/Fragment Algorithm

There are several pro-forms other than pronouns. These forms may be amenable to the use of the pronoun/noun phrase elliptical fragment algorithm, with modifications for part of speech. This would also have to include separate foci for each additional part of speech form/fragment combination.

The part of speech most suitable for this is the verb phrase. The pro-verbs for abbreviated reference are the forms of "to do". For example, "We like to play outside. John does too.", and "Whenever we play outside, John does too.", where "does" refers to "like to play outside" and "play outside", respectively. Notice that the verb in the referent verb phrase does not agree in number with "John", though.

Other pro-forms include pro-adjectives ("such"), pro-actions ("do it", "do so"), and pro-sentences ("it", "such"). Each of these forms has different exceptional cases and would require individual study.

A generalization of the elliptical handling portion for noun phrases to each part of speech could be made with less effort. This would retain foci for each selected part of speech and perform salience weight upkeep on the concepts represented in these foci. This may be the preferred solution for commonly occuring non-sentence elliptical input, such as prepositional phrases.

6.9.3 Greater Semantic Knowledge

Three types of anaphora are mentioned here as being very difficult areas. These are noun phrases that could be independently interpreted, but are instead being used for referring to other noun phrases that have occurred elsewhere in

discourse. It may be debatable in some cases whether this usage constitutes "abbreviated reference", since some of these reference types add information about the referent object.

Definite reference is the name given to the use of a definite noun phrase ("the ball") to refer to a specific object within the focus of the discourse ("we were playing football"). This type of anaphora certainly needs "immediately associated" rules. It also raises questions about natural language systems with regard to intentionality and extensionality.

Some progress may be made in this area by noting that in a context of discussion about "the Alamo" (a ship) the following two sentences yield the same results: "What is in the aft hold?" and "What is in its aft hold?"

Similar to the definite reference problem is reference to a specific noun phrase by paraphrases and epithets. A full processing of these types must include adding the information about the referent that is given in the reference.

6.10 Summary: The Nature of Reference

The problem of reference will always be very important to the design and implementation of natural language systems. This is because the semantics of a sentence is what the sentence refers to, be that things or actions, real or imagined. A natural language knowledge base system can only refer to elements in the knowledge base, except to add data or for fill-in-the-blank diagnostics about what is not found. Study of some of the forms of abbreviated reference found in human communication reveal a great distance between human understanding of a situation and the formalized systems that

have been devised for interpreting natural language. Anaphora would be much easier to resolve if these systems "knew what we were talking about" before an abbreviated reference is made.

Chapter 7: Corrections

7.1 Introduction

Several kinds of corrections are attempted by the portion of the system for processing ill-formed input. Ill-formed input for the purpose of this chapter is defined to be input for which there is no arc covering the whole input, or input with a non-noun phrase or non-sentence arc wholly covering the input without any elliptical resolutions. Ill-formed input requires diagnosis to aid the user in forming well-formed input, and also to indicate that any processing that has taken place was not exactly as originally requested.

In this chapter the corrections that are attempted will be described. Towards the end a comparison with treatments of ill-formed input by other systems will be made. The vocabulary diagnostic and diagnostics for explaining changes to user input are also given here. The final (syntactic) diagnostic requires some discussion of its own, and will be considered in the next chapter.

A primary distinction in the corrections attempted is made between major and minor corrections. Minor corrections are tried first. Only two submissions for evaluation are required because most of these corrections can occur simultaneously. Major corrections are a group of other corrections, each with essentially its own evaluation because little interaction is possible among them.

Examples of Minor Errors: "what are ships?"

```
what afe ships?
whatare ships?
OK, what are ships?
what are ships????????
what are shir ships?
```

Examples of Major Errors: "what are ships?"

what is ships?
what rare ships?
what ships are?
is what are ships?

The correction process is initiated whenever the conditions indicate that the input is ill-formed. This is either when the whole arcs program finds no whole arcs, or when whole arcs found that were neither sentences nor noun phrases fail to resolve as elliptical fragments. The correction process terminates either when some modification to the original input is found that evaluates correctly, or when a syntactic diagnostic is found that sufficiently diagnoses the situation.

The actual correction process normally consists of placing new arc(s) into the parsing chart after the parsing chart has been generated by the preprocessor. The form of the new arc(s) will be indicated in the examples by the arc(s) over the ill-formed input.

7.2 Minor Corrections

Corrections to be made by the system that will be considered minor are: the spelling corrections, and the deletion of several usages of words and punctuation occurring in input that can be interpreted as phatic. Because spelling correction will initially be invoked by words found that are either not in the lexicon, or not covered by the application of grammatical rules, the diagnosis of vocabulary errors is a natural byproduct of spelling correction attempts. Thus although these corrections are minor in the sense of the changes made to the original input, they treat the majority of input errors found in the protocols; vocabulary, spelling and punctuation errors

accounted for a total of two-thirds of the error fragments found.

First the individual mechanisms for making the small corrections will be described. Then how they fit together in the evaluation process will be covered. A discussion on the syntactic diagnostic potentially resulting from the minor corrections programs finishes the description of the treatment of these types of errors.

7.2.1 Spelling

Any string of letters (possibly with numbers) separated by other delimiting symbols (such as punctuation or spaces) is made into an arc of the parsing chart with the part of speech "identifier" by the preprocessor, if the string was not found in the lexicon. These strings will be called identifiers in the sequel. The spelling corrector operates on identifiers which have not been covered by any other arc in the parsing chart. After the parser has applied the rules of grammar to the input and the input has been determined to be ill-formed, the spelling corrector attempts to match the strings of uncovered identifiers to the words in the lexicon. Thus spelling correction is a form of the approximate string matching problem.

Two types of spelling corrections take place within the minor correction stage of ill-formed input processing. These correct errors within words and errors between words. This distinction is convenient because of the preprocessor's design; corrections of missing space errors may be made as the words are being looked up in the lexicon. Another kind of spelling correction will be dealt with in the section on major corrections.

7.2.1.1 Within Words

A Damerau-Levenshtein kind of metric [Hall 80] is used as the measure of string similarity. Thus a given string is an approximation of another if modification of the string by editing changes of one addition, one deletion, one substitution, or one transposition of adjacent letters creates an exact match. (If modifications of more than one editing operation are considered, the operations should be limited to additions and deletions, as the others are combinations of these). The spelling correction algorithm thus must generate all strings found in the lexicon that are similar to the uncovered identifier. This collection of strings is referred to as the neighborhood of the spelling correction candidate. Since the neighborhood is highly dependent on the lexicon, the string manipulation portion of the mechanism produces a set of strings in which the neighborhood is contained, rather than the neighborhood itself. Then the lexical look-up routines are used to reduce this set to the neighborhood.

Bounds can be given for the size of the set of strings for lexical look-up. For an identifier of length n and a language with an alphabet (number of characters) of size k, this set will have at most M elements, where:

$$M \leq (n+1)k + (n) + n(k-1) + (n-1)$$

$$= (\text{additions}) + (\text{deletions}) + (\text{substitutions}) + (\text{transpositions}).$$

It is extremely rare that this maximum will be attained. If every one letter addition, deletion, substitution, and transposition were attempted, adjacent letters could not be the same to achieve this maximum. The lexical look-up method

further eliminates many of the unnecessary elements of this set.

The lexicon contains the information necessary to build the arcs corresponding to the words of the input. This information is accessed by passing through two tables of Boolean values. A sequence of characters is "hashed" by a function on the characters that applies to one letter at a time, from left to right. This function gives a unique result for each (finite) sequence of letters, which is then used as the index to the two Boolean tables. The first table indicates whether or not the value of the hashing function on the sequence of characters is equal to the value of the hashing function on the initial portion of some word in the lexicon. The second table indicates whether or not the value of the hashing function on the sequence of characters is equal to the value of the hashing function on some word that actually is in the lexicon. The information about how to complete the arc(s) corresponding to that word can then be found by completing the hashing method, which will not be described here. A few selected strings illustrate this:

String:	Hashing Function:	First Table:	Second Table:	Corresponding Arc(s):
qq	h(qq)	false	false	---
Joh	h(Joh)	true	false	---
John	h(John)	true	true	|John| (noun)
shi	h(shi)	true	false	---
ship	h(ship)	true	true	|ship| (noun)
(note lexical ambiguity)				|ship| (verb)

This method eliminates many unnecessary elements of the set for lexical look-up because modification of the spelling correction candidate proceeds from left to right. Thus it can

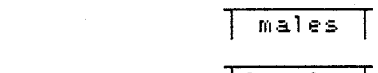
stop when the first table indicates that the portion of the string before the point of modification is not in the lexicon. No modification beyond this point will result in corrections. An example using deletion as the only possible modification helps to clarify this process: (|X| deletes character X)

Candidate:	b f e a r	
Attempts:	|b|f e a r	Found: fear
	b|f|e a r	Found: bear
Stops Before:	b f|e|a r	

(as "bf" is not the initial sequence of any lexical word).

This eliminates any modification attempts after one error has already been detected. Therefore both the look-up process and the string manipulation portion are kept from wasted effort by the first hash table. Of course this curtailed set still includes the neighborhood of the uncovered identifier.

After neighborhood generation, the arcs of the words in the neighborhood are placed in the parsing chart above the uncovered identifier. Spelling correction within words is complete when this has been finished for each candidate. The parser will use these arcs to form all possible grammatical combinations of the input and its modifications.

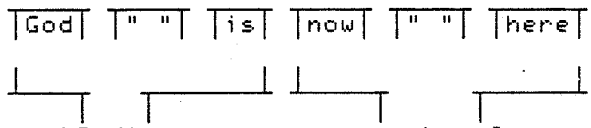



```
>list   emales .
 The following word is not in the vocabulary: emales
 Corrections:
 1)list females.
   Joan
   Jean
 2)list males.
   Bill
   John
```

7.2.1.2 Between Words

In the normal operation of the preprocessor, which assumes well-formed input, arcs are put into the parsing chart wherever a string has been found in the final segment hashing table. Sometimes this may be within an identifier, so that prefixes may be accepted: "milli", "centi", etc., are recognized this way. Hashing will begin again at the end of the prefix, after hashing is complete on the current identifier. Thus all ambiguous interpretations are obtained. For "picometer", the arcs for "pi", "pico", and "meter" are all generated. (The "picometer" arc is generated from a grammar rule by the parser).

Execution is different when the preprocessor is called after the input has been determined to be ill-formed. When hashing indicates the end of a word which is not at the end of an identifier, the arc for a space is put into the parsing chart after the initial word. This is done in a way that appears as optional to the parser, so that this modification will not interfere if it is not a correction. Insertion of a space may create ambiguities:




```
>Godis              nowhere?
 The following word is not in the vocabulary: Godis
 Corrections:
 There are 2 answers:
 1)God is nowhere?
   no
 2)God is now here?
   yes
```

(This example is attributed to an atheist, who wrote a sign similar to the input, intending the first interpretation. His young child promptly read the second [Osborne 77, p. 47]).

Ambiguities of this sort are no problem, as the interpretations are clear from the echoes of the executed input. Multiple spelling corrections within words are diagnosed the same way.

7.2.2 Phatic Words, Phatic and Erroneous Punctuation

Treatment of phatics occurring in user input is also considered a minor correction. Individual words or phrases that are phatic have already been described in chapter 1. Phatic punctuation has the same function; repeated punctuation and other symbol strings ("!@#$%^&*") were observed in the terminal-to-terminal mode being used as phatics. It was further pointed out in chapter 2, that although not precisely correct, ignoring phatics is very close to the proper interpretation of these kinds of fragments.

7.2.2.1 Optional Deletion

Ignoring of phatics is accomplished by modifying the parsing chart. Arcs are inserted over the deletion candidates (the phatic arcs). These deleting arcs may be either space arcs or other delimiting symbols used at the boundary of the phatic. Modification takes place in a fully ambiguous manner, and the parser has the option of using the phatics so modified in any combination, including using none of them. This is important because phatic words will often have other interpretations.

Deletion of phatics receives no diagnosis except in the presence of other errors or ambiguous input. These examples illustrate this:

```
  |" "|                  |" "|
> OK,  is John    ok ?
  yes
```

```
 |" "|    |John| |" "|
> ok.  Is  Yohn    ok ?
 The following word is not in the vocabulary: Yohn
 Correction:
 Is John ok?
 yes

 |  " "  |
> well,  who is Smith?
 There are 2 answers:
 1)who is Smith?
   "Smith" defined as "Bill Smith"
   Bill Smith
 2)who is Smith?
   "Smith" defined as "John Smith"
   John Smith
```

This ability to optionally delete arcs from the parsing chart forms the basis for some other corrections. Words or phrases can be made phatic, and then the deleting program will do the rest. A minor correction that uses this capability is the deletion of spelling correction candidates.

The deletion procedure can also (optionally) delete all punctuation. Coupled with robust grammar rules for punctuation, this provides for the correction of most punctuation errors (91% of those in the protocols).

7.2.3 Evaluation with Minor Corrections

Two evaluation equivalents may take place within the minor corrections portion of the system. These consist of parser and semantics calls after suitable modification of the input. The first group of attempted changes consists of spelling correction both within and between words, and deletion of lexically phatic words and phatic punctuation. This produces a parsing chart of the form of preprocessor output. This is submitted to the parser. After parsing, the whole arcs routine determines the well-formedness of the result(s) and calls anaphora resolution and ambiguity diagnosis as necessary. Good output from whole

arcs is evaluated semantically.

Before phatic deletion within the first evaluation, the parsing chart with both kinds of spelling corrections is saved. If the first attempt has failed either semantically (a semantic diagnostic is not considered bad output for a correction) or syntactically, this saved parsing chart is modified differently for a second try. New modifications are the optional deletions of phatics as before, along with all punctuation and spelling corrector candidates. Evaluation then proceeds as before.

7.2.4 Minor Error Diagnosis

A few examples have been given of the syntactic diagnostic indicating a word outside the system's vocabulary. This is slightly different for more than one word:

```
 |who|  |is|  |John|
> woh   si   Yohn ?
 The following words are not in the vocabulary:
   woh si Yohn
 Correction:
 who is John?
 John
```

If the minor corrections do not produce any good results or semantic diagnostics, then control would normally pass to the routine supervising the major corrections. The exception is when there is a word not in the vocabulary.

Processing terminates after trial of the minor corrections when a vocabulary diagnostic may be given. There are several reasons for this. The vocabulary diagnostic gives the user concrete help in understanding the problem. Also, if other errors are involved, correction of this error must take place before further diagnosis. True vocabulary errors are far more common (34% of input errors in the protocols) than the spelling errors that will not be corrected by the routines presented here

(3% of the input errors). Thus a spelling corrector with a more relaxed definition of similarity will not be a part of the major corrections, because of the rarity of its usefulness and the previous diagnosis of the problem. (Within the major corrections, it may therefore be assumed that all words in the input are in the vocabulary). For all these reasons, no further processing after attempting the minor corrections will be performed on input requiring a vocabulary diagnostic.

7.3 Major Corrections

The major corrections are more concerned with the words of the input rather than the characters. All words in the input are in the vocabulary at this stage, thus the problem has more to do with the grammar. Most of the major corrections substitute a word for one in the input. The following program descriptions explain how the substitute words are found. These words are put into the parsing chart the same way that the possible spelling corrections were earlier. The other major corrections tried are word deletions. After describing the individual pieces for major corrections, it will be explained how they fit together.

7.3.1 Rule Relaxation

The first word substitutions come under the heading of rule relaxation because these substitutions constitute relaxations of the requirements of selected grammar rules. Instead of relaxing the rules during their application, word substitution is done because of the need to redo all parsing above (after) the application of the relaxed rule, and for the sake of improved diagnostics. Thus an echo of the modified input is used as the

explanatory diagnostic, rather than a message explaining which tests were violated (as in [Kwasny 80, 81], where many of the ideas for rule relaxation come from).

Two general relaxation methods are exemplified here by two procedures that find words for substitution. These were the ones chosen for implementation, both as examples of this type of correction and because of their apparent usefulness. The task of filling out the rest of this meta-grammatical portion of the system needs the results of protocol analysis that is not plagued by errors subject to the minor corrections.

7.3.1.1 Different Forms of the Same Word

There are many words that have a recognizable root. This can be seen in the conjugations and declensions of verbs and nouns. By saving the information on selected commonly encountered roots, one word having a given root can be expanded to all the words with that root. If one word occurring in ill-formed input can be used to generate this new kind of neighborhood, then the addition of these other forms to the parsing chart will produce the equivalent of relaxation of various feature tests on grammar rules.

The example given here is of relaxation to all forms of the verb "to be". Thus if number or person features have impeded parsing of the input, these requirements will be relaxed by these substitutions:

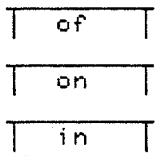
```
     | am|

     | is|
>who  are  Bill?
 No sentence was found including the whole input.
 Correction:
 who is Bill?
 Bill
```

Another word that would be good for this kind of relaxation is the verb "to have". These two verbs have a special place within the ASK System, as they are used for defining other verbs. Thus these two relaxations would be more useful than ones for other verbs.

7.3.1.2 Different Words with the Same Part of Speech

A few parts of speech have a limited number of elements. For a word with one of these parts of speech, relaxations may be made to each member of that part of speech. The example of this kind of part of speech is preposition. When other prepositions are added to the parsing chart where one was already located, any parsing blockages due to incorrect usage of prepositions will be avoided:




```
>who is the parent  between  John?
 No sentence was found including the whole input.
 Correction:
 who is the parent of John?
 Martha
```

Limited subsets of the part of speech pronoun may also be good for this type of relaxation, to cover pronoun usage violating animate, gender, or number requirements.

Within the present structure of the corrections performed by the system, it is proper to perform all of these rule relaxations together. Since few other rules are likely to be applied as often as these for English (used with native speakers), ambiguities resulting from the interaction of these corrections will be within reason. All possible corrections

will be found this way, to demonstrate how good sentences may be formed.

The rules given for relaxation here are dependent on the use of English as the natural language. It is hoped that these examples will translate well into other base languages, or at least serve to give the idea of what kinds of changes can be appropriately made at this stage of execution.

7.3.2 Spelling, Again

Another major correction is the second level spelling corrector. Words from input reaching this point are already known to be in the vocabulary. Spelling errors do not always remove words from the vocabulary, though. Grammatical errors that are really spelling errors come about this way. Second level spelling correction is begun by a routine that makes all words of the input satisfy the uncovered identifier property. Then spelling correction proceeds as before, this time generating the neighborhood around each word of the input. The diagnostics given for this and the remaining major corrections are of the same form as those for the rule relaxations:

```
    |on|
    |if|
    |is|
>who  in  John?
 No sentence was found including the whole input.
 Correction:
 who is John?
 John
```

7.3.3 Word Order

Changes in order of words of the original input is accomplished by placing the arc of each word over each other

word. This makes for all possible combinations of word order after parsing. Fortunately, the evaluation's syntactic and semantic considerations will eliminate many of the generated permutations. This is one of the more radical changes within the major corrections.

```
      | is |  |John|
>who  John    is  ?
 No sentence was found including the whole input.
 Correction:
 who is John?
 John
```

7.3.4 Deletion, Again

The most radical change that is attempted is the optional "deletion" of each word in the input. Deletion takes place in a manner that the parser sees as optional, so that the parser will try each combination of the words of the input, with no change in the order of the words. This is considered the greatest change necessary because it will always obtain at least the noun phrases, for which a reasonable response can be given. Bordering on too much modification, this change is retained primarily to show the user that something can be done with almost any input. Processing of the noun phrases will put them on the potential referent list, so that they may be referred to in the next input.

```
  |" "|
> is,  who is John?
 No sentence was found including the whole input.
 Correction:
 who is John?
 John
```

7.3.5 Evaluation with Major Corrections

Four evaluation equivalents are possible within the major corrections portion of the system. These apply different sets

of correctors to preprocessor output before submitting the parsing chart to the parser and the whole.arcs programs, and the semantics program if it passes these. The correctors are attempted in order of increasing degree of change to the original input: first the rule relaxations, then the second level spelling corrector, the word permutator, and the word deletor. Processing stops whenever a correction is found, retaining the full ambiguity at that level. It also stops when there are no more corrections left to try.

Whenever ill-formed input processing reaches the major corrections, the diagnostic "No sentence was found including the whole input." is put on before any corrections. The next chapter discusses the suitability of this syntactic diagnostic, and the optional maximal covers diagnostic for further information about reasons for ill-formedness.

7.4 Corrections Made in Other Systems

Several of the corrections made by the ASK System were motivated by the study of other systems, both for natural language processing and for word processing. There have also been a few recent papers that discuss similar issues, reflecting independent work.

Word processing systems have contributed some ideas for spelling correction. [Damerau 64] implemented a spelling corrector producing the four basic changes to input strings. In [Morgan 70], the notion of using spelling correction with system programs was discussed. Implementation of a program for spelling correction was the topic of [Peterson 80], as an example of an experiment in programing design. This final

report also has an annotated bibliography for reference to other literature.

There are several novel ideas for spelling correction that were ruled out. The use of stored information about n-grams, for the possibility of occurrence of sequences of letters in English, as in [Riseman 74], was not used. This is because it is desirable that the system correct strings representing variables, which are often chosen because they do not occur in English. The method of [Mor 82], of saving the neighborhood of words (the common spelling errors themselves) in the lexicon, would put extreme demands on the size of the lexicon.

The ill-formed input methods described by Weischedel, Sondheimer, and Kwasny have influenced the rule relaxation portion of the system. The rule-based framework for processing errors in grammar of the rule relaxation form was reported in [Weischedel 81], [Sondheimer 80], and is rooted in some early work [Weischedel 78]. Several rules to relax were analysed in [Kwasny 80, 81], motivating the rule relaxation examples given earlier.

There are two primary systems that have developed in parallel to the ASK System and provide some of the correctional and habitability capabilities. The first of these is the flexible parsing work of Hayes [Hayes 81b]. Based on the ideas of graceful interaction presented in [Hayes 79], system correction of a variety of errors takes place within the parsing method. These corrections include spelling correction within and between words, some processing of fragments, and selective deletion in particular cases.

The other system with an overall approach to ill-formed input processing is the error tolerant analysis (ETA-Interface) of [Bradford 82a, 82b]. This does insertions, deletions, and rearrangements of words in an input. A neighborhood of grammatical sentences is generated from these changes. These are presented to the user individually for confirmation. The distance between English sentences is defined, and corrections proceed in the order of increasing amount of change to the input. The system does not appear to have a notion of ambiguity, which is implied by modifications of equal distances.

7.5 Final Comments

The evaluation order of the minor and major corrections completes the description of the progression of modifications to user input that was begun in the previous chapter. The implied measure of editing changes, depending on considerations from the study of protocols and syntax, will be discussed more fully in chapter 9. The evaluation method shows how the system uses both syntax and semantics to make corrections. The ambiguity capabilities of the system are used on each of the levels of corrections, to provide an overall structure to the partial order of changes made to ill-formed input.

Chapter 8: Final Diagnostic

8.1 The Paradox of Diagnosis

How can ill-formed input that does not give a vocabulary diagnostic be diagnosed habitably? The need for diagnosis assumes failure to process the input to completion, so that the input may have been either outside the system's grammar or outside the grammar of the natural language itself. In either instance, the original input should exceed the system's grammar, although perhaps not the meta-grammar, which is the possible corrections that are to be attempted. Input beyond the system's grammar proper should be diagnosed. Input within the meta-grammar can be corrected as well as diagnosed. The input beyond the meta-grammar can only be diagnosed, which may be limited to simply the recognition that it is beyond the system's capabilities.

Diagnosis attempts to explain more than just that the input was ill-formed. Ideally, diagnostics should be helpful in forming well-formed, complete input. Therefore, the ideal diagnostic is closely related to a kind of correction.

The paradox of these considerations is this: if the system either knows or can suggest a method of modification to the user's input that might produce well-formed (and complete) input, then why not have the system try the modification itself? This is the problem behind such diagnostics as:

```
>who is John
 syntax error: question mark expected
```

Clearly, system replies of this character are not within what would be considered habitable interaction. But this diagnostic

does indeed accurately describe the problem, and suggest the appropriate method for its correction.

The paradox of diagnosis is a strong motivation for handling anaphora and making corrections in the ill-formed and incomplete input encountered by natural language systems. However, processing of anaphora and corrections make the diagnostic problem even more difficult: what can be suggested to the user if all attempts at correction have failed? The approach taken here is to give a short, quick diagnostic in all cases, and to provide an optional deeper syntactic analysis of the input if the user desires more information. The first diagnostic will not interfere with users familiar with the system, and will notify all users of the further diagnosis available. This deeper analysis is called the maximal covers analysis diagnostic.

The primary exception to otherwise futile diagnostics is the vocabulary diagnostic. It is an example of a diagnostic that isolates the problematic portion of the input, and the user can be expected to try synonyms, or a different approach. Either way, the word in violation should not be repeated by intelligent users. Some other systems may try to use synonyms as a correction, but if there are synonyms available in the system then the troublesome word was in the vocabulary. Spelling correction and deletion are the only recourse for words outside the vocabulary (other than more general substitutions), but the diagnostic is worthwhile if these attempts fail.

The response to pronouns with no resolutions is another exception. These are the only two exceptions within the system design. Input that fails all attempts at correction and

anaphoric resolution and cannot be given either of these diagnostics will receive the final diagnostic: "No sentence was found including the whole input." (This will be called the no whole-sentence diagnostic). The maximal covers analysis will be available for any input producing the no whole-sentence diagnostic.

8.2 Generalized Deepest Parses Heuristic

The maximal covers analysis of ill-formed input is based on the deepest parse heuristic of [Weischedel 80], which is an outgrowth of ideas in [Weischedel 77]. The deepest parse of the input is found as a basis for diagnosis. A report on test cases in the first paper showed that over 90% did stop parsing at the point of difficulty. But this was only 36 test cases, and as reported in [Kwasny 80, p. 65] there are problematic cases. The ATN formalism, at least as conventially implemented, is biased towards left to right parses. The particular method in the first paper also ignores the fragments not in the deepest parse.

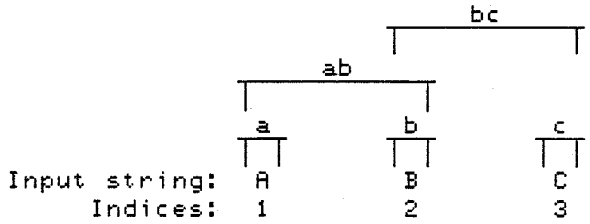
The deepest parse heuristic is expanded in the maximal covers analysis to account for the problematic cases and to eliminate the bias towards left to right parses. Ambiguity is used as a tool for presenting multiple equally likely deepest parses. The remaining portions of input not within one of the deepest parses will be subject to the same analysis.

8.3 Definition of a Maximal Cover

The description of the maximal covers diagnostic and its generation depends on the formal definition of a maximal cover. The maximal covers diagnostic consists of the listing of each

maximal cover over the input. The terms maximal cover and maximal covering will be used interchangeably. The definition proceeds by defining: arc, range, containment, maximal arc, cover or covering, and maximal cover or maximal covering.

These definitions are illustrated in the following example. Here, the lower case letters refer to names of the arcs, and upper case letters are elements of the input string. It will always be assumed that the input string is finite.



Definition: Arc: An arc over a portion of the input is any of the arcs of the parsing chart. This includes the arc's information about part of speech, syntactic features, and literal strings. (Each of the individual elements of the input string constitutes an arc, whose part of speech is the character if no other is given).

Arcs: a, b, c, ab, bc.

Definition: Range: An arc x has a range of [n,m], where n and m are (integer) indices into the input string, (n <= m), if x is an arc over the input string from n to m.

Ranges:	arc:	range:
	a	[1,1]
	b	[2,2]
	c	[3,3]
	ab	[1,2]
	bc	[2,3]

Definition: Containment: An arc x is contained in another arc y if the range of x is a proper subset of the range of y.

Containment:

ab contains both a and b
bc contains both b and c

Definition: Maximal Arc: An arc x is a maximal arc if there is no arc y containing x.

Maximal Arcs: ab, bc.

Definition: Cover, or Covering: A set of arcs is a covering of an input string if the ranges of the arcs are mutually disjoint, and the union of the ranges includes the indices of every element of the input string.

Covers: (a,b,c), (ab,c), (a,bc).

Definition: Maximal Cover, or Maximal Covering: A covering of an input string is a maximal covering if no non-maximal arc appearing in the covering is contained in a maximal arc whose range does not intersect a maximal arc in the covering.

Maximal Covers: (ab,c), (a,bc).

These definitions make specific the intuitive notions relating to a deepest parse for the ASK System environment. A parse is an arc; a deepest parse is a maximal arc; a cover is a set of parses including the whole input; and a maximal cover is a cover containing as many deepest parses as possible after the choice of one of the deepest parses. The complete set of maximal coverings will have each maximal arc represented in at least one of the coverings.

Since each element of the input string is an arc, there will always be a covering of the input string (just the collection of these arcs). This further guarantees that there will always be a maximal cover:

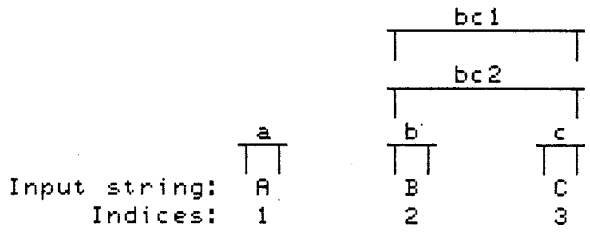
Proposition: If there is a covering over an input string, then there is a maximal covering.

Proof: Suppose there is a covering Z that contains an arc a violating the maximality condition, i.e., there is a maximal arc m such that a is contained in m and m intersects none of the maximal arcs of Z (if there is none then Z is maximal already). It may be assumed that m is not in Z, since if it was then Z would not be a covering because m intersect a is non-zero.

Thus Z′ can be formed by adding m to Z and removing all elements of Z that are contained in m. If there are no remaining arcs violating the maximality condition, then Z′ is a maximal covering. Otherwise, this process may be repeated until a covering is found that is maximal. The process must stop because the input string is finite. (End of proof)

This shows that the search for maximal covers is not in vain. Although the proof could be used as a basis for forming maximal covers, the actual algorithm to be described will work differently. It starts with the maximal arcs and fills in with the others where necessary. Thus the large chunks are covered quickly, and the algorithm works more quickly than the proof's approach.

The set of maximal coverings is ambiguous if there is more than one maximal covering. Ambiguity may arise for either structural or lexical reasons. The example above illustrates a structural ambiguity. The following one shows a purely lexically ambiguous set of maximal coverings:



Maximal Covers: (a,bc1), (a,bc2)

8.4 Description of the Algorithm

There are three major parts of the algorithm for preparing the maximal covers analysis diagnostic. These are: finding arcs representing the ranges found in the parsing chart, finding the forms corresponding to the structural ambiguities of the set of

maximal coverings, and preparing the output for the user by adding the lexical interpretations of the arcs into these structural forms and adding explanatory messages.

The first and third portions are quite simple and will not receive as much discussion. The first part just looks through the parsing chart to find what ranges exist that are covered by arcs. The range information is essentially contained in the arcs, so that this routine only needs to observe each arc. This information is then available to the second step, so that the algorithm for that portion may ask if an arc with a given range is in the parsing chart.

8.4.1 Finding Structural Ambiguities

The major section of the algorithm finds the forms of the structural ambiguities of the set of maximal coverings. A single structural ambiguity of the set of maximal coverings is just the set of ranges that forms a maximal covering. An array representation of the ranges of arcs found in the input will be useful for the algorithm's description. This is the array for the structurally ambiguous example earlier:

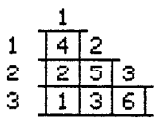
	1		
1	X	2	
2	X	X	3
3		X	X

The numbers over the columns represent the initial number of a range, and the numbers of the rows are for the final number of a range. An X in a location of the array indicates that an arc exists with the range from the initial number to the final number, with the range from the column number to the row number.

Notice that an X in the lower left-hand corner indicates that there exists a whole arc covering the input. An X in any

location indicates an arc containing all the arcs represented by the X's either above in the same column, to the right in the same row, or both in a row above and a column to the right of that position.

This array is searched by the algorithm starting at the lower left-hand corner, and proceeding down the diagonal (upper left to bottom right) immediately above the starting point, and then proceeding to successively higher diagonals. In this manner the maximal arcs are found first in the process of constructing maximal covers. This diagram indicates the order of search:



The search continues until every range represented in the diagram has been searched or contained in an arc found in prior search. Ranges are eliminated from the search by arcs containing those ranges or recursive calls on those ranges. Thus all the ranges found in the search correspond to maximal arcs.

When a maximal arc is found, a maximal cover is developed by applying the algorithm recursively with the starting point at the range(s) that would complete the cover. Thus for an input string of length 3 with no whole arcs, finding a range of [1,2] will invoke a search for "relatively" maximal coverings over the range [3,3]. (After this, the range [2,3] must be searched before all the ranges will have been attempted). At most two other sub-searches are required, for any given level. For instance, two would be necessary in a string of length 5 where the range [2,4] was found first. This would initiate further searches in the ranges [1,1] and [5,5]. Ranges searched by

recursive calls are eliminated from the ranges for searching within the calling routine, because this sub-portion will have already been covered by the returned (sub-) maximal covering.

If a structurally ambiguous sub-covering is returned, it is checked to make sure that each arc of the sub-covering that is not maximal does not violate the maximality condition. Non-maximal arcs of the covering may not be contained in maximal arcs not intersecting any other maximal arc of the covering. Any violating sub-coverings are removed.

The algorithm can be proven correct by course of values induction on m, the number of elements in the input string:

Theorem: The algorithm finds all and only the maximal structural coverings.

Proof: For m=1, the only range an arc may have is [1,1], thus this is the only maximal structural covering. The algorithm first searches for [1,1], finds it because the definitions guarantee its existence, and terminates. Thus it is correct for n=1.

For m=n, n > 1, assume that the theorem holds for all k, where n > k >= 1. The algorithm begins its search. If [1,n] is found, this is the only maximal structural covering; the process stops, as all other arcs are contained in this first one, and the proof is complete. If [1,n] is not found the search continues. Suppose [i,j] is the first range found. This is a maximal arc, because no arc was encountered before it. It eliminates searching in the ranges contained in [i,j]. The program is recursively called for the range [1,i-1] (if i is not equal to 1), and for the range [j+1,n] (if j is not equal to n).

Since these ranges correspond to input strings with lengths less than n, all and only the (sub-) maximal coverings

will be returned, by the induction hypothesis. Any structural ambiguities that are not maximal at the higher level are eliminated. The program will not search again in the ranges contained in the ranges used in the recursive call(s). The sub-maximal structural coverings remaining are combined with [i,j] to form maximal structural coverings. The same arguments hold for the other [i,j] ranges found in the remaining search. Thus only maximal structural covers will be found.

To show that all maximal structural covers are found, notice that only maximal arcs are found in the search, and are combined with all possible combinations of other maximal arcs by the recursive call(s). All possible combinations of filling with non-maximal arcs are also added by these calls, and eliminated where maximality on the higher level is violated. Thus all and only maximal structural coverings are found. (End of proof)

8.4.2 The Diagnostic as Output

The final section of the algorithm is for preparing the output from the list of structural ambiguities. Underlying structural ambiguities are left that way to avoid repeating the arcs in the output. The lexical ambiguities of the parsing chart are added as underlying ambiguities to the structural forms.

The diagnostic indicates the relative structure of arcs by indentation. For each range of arc occurring in a structural ambiguity, the fragment of the input string indicated by that range is used as a label for those arcs. All arcs are individually labeled with their part of speech, and this serves to disambiguate arcs with identical ranges. Some elimination of

lexical ambiguities takes place when one part of speech indicates deeper parsing. For example, nouns underlying noun phrases with the same range are suppressed from output. This example illustrates several of these points. (Corrections taking place are not shown here).

```
>is the John?
 No sentence was found including the whole input.
 The following fragments were identified:
 Fragments of: "is the John?"
   Ambiguous
    the fragment:"is the John"
     Ambiguous
      has part of speech: verb-phrase
      has part of speech: verb-phrase
    the fragment:"?"
     has part of speech: pct
   Ambiguous
    the fragment:"is"
     has part of speech: verb-phrase
    the fragment:"the John?"
     has part of speech: sentence
```

The indices into the input string are given for reference:

```
>  i  s     t  h  e     J  o  h  n  ?
   1  2  3  4  5  6  7  8  9 10 11 12
```

This example has a structurally ambiguous set of maximal covers. The maximal covers in structural form are: ⟨[1,11], [12,12]⟩, and ⟨[1,2], [3,3], [4,12]⟩. The range [1,11] is lexically ambiguous, although the ambiguity is in the features, so it is not differentiated. The range [1,2] is also, but the arc with part of speech copula was eliminated because verb phrase is treated as a deeper parse. The space character [3,3] in the second structural maximal cover was not included in the output. The different structural ambiguities are prefixed by the "Ambiguous" message. This also precedes lexically ambiguous ranges. Notice that there was a sentence in the input: "the John?". The major correction that optionally deletes words will

try this and respond accordingly, but it by no means constitutes full processing of the input.

8.5 Usefulness of the Maximal Covers Diagnostic

Is it worth the effort to produce the maximal covers diagnostic? Some linguistic knowledge is required for it to be valuable. (This brings up another paradox: the people who need it least, understand it best). It is helpful in debugging grammar specifications. It is in some sense the best that can be done with the situation. It indicates the extent to which the input was processed, and locates the point of blocking in the parsing process (often the point of difficulty in the input). No suggestions can be made that have not already been attempted by the system.

The decision was made to retain the capability to produce the maximal covers diagnostic, but to make it optional, as the individual user decides. The user may specify:

```
     >maximal covers analysis
      maximal covers analysis has been turned on.
or
     >maximal covers analysis on
      maximal covers analysis has been turned on.
or
     >maximal covers analysis off
      maximal covers analysis has been turned off.
```

with the expected results. The full diagnostic will follow the no whole-sentence diagnostic when these messages are desired.

Chapter 9: Final Topics and Conclusions

9.1 Summary

The need for improved habitability has precipitated several active research areas within the Computer Science field. This thesis represents contributions to most of these areas pertaining to the remaining questions of habitability. Studies were carried out observing human performance in an interactive setting to determine the most necessary constructs to process for general habitability improvement. An overall structure was proposed for the coordination of all these new constructs, and equally likely different interpretations of input. The interaction of different kinds of changes made to input is incorporated into this structure. Advances were made in the processing of each of the following types of input: ambiguous, anaphoral, ungrammatical, extra-grammatical. A theory of diagnosis was devised for the overall structure, performing natural diagnosis on any input requiring processing beyond the explicit input request. A diagnostic was proposed for the explanation of fragmentary input for which no recommendations for correction can be found. All of the structure and algorithms developed have been implemented as a part of the ASK System.

9.2 Conclusions

Four topics are considered in conclusion. The results of the implemented system as applied to the initial human-to-computer protocols gives an approximate evaluation of the improved habitability of the system. The metric on possible input strings can now be developed more fully, since each of the input

modifications that might take place have been described. Finally, some discussion is presented about the future work and more thorough habitability evaluations that continue this path of research.

9.2.1 Results: Implied Statistics on Old Protocols

Table 9.1 presents the results of the Robust Sentence Analysis system within the ASK System as applied to the initial human-to-computer protocols. (Percentages in the last column refer to the percentage of that error fragment corrected, not to the percentage of error fragments.)

Table 9.1: Error Fragments: Corrections

	Total		Corrected	
Totals	453		287	63%
Vocabulary	153	34%	68	44%
Punctuation	79	17%	72	91%
Syntax	69	15%	47	68%
Spelling	66	15%	53	80%
Transmission	33	7%	33	100%
Definition Format	25	6%	?	
Unexpected Input	14	3%	14	100%
Bug	14	3%	?	

Some comments are necessary about these results. Robust Sentence Analysis has concentrated on the first four fragment types. The others are more a function of the system. The occurrence of bugs and definition format errors cannot be predicted. Some improvement can be expected in the number of definition format errors because of new forms that are available. It is hoped that there will be no bugs. Unexpected input errors are avoided by system changes for defaulting. Transmission errors were errors on the part of the experimenter, not the user's input.

Vocabulary errors are not exactly corrected, but it was assumed that after a vocabulary diagnostic the same vocabulary

error would not repeat again within the same context. All vocabulary errors would be diagnosed, as would all spelling errors that would not be corrected by the system. Punctuation errors that were not corrected consist mainly of places where added punctuation was necessary, rather than less. This is seen to be a much less frequent case.

The number of syntax errors corrected does not reflect all the forms that are strictly speaking syntax errors that would be corrected. Two of the false starts would be corrected, for a total of 8% of these forms. Of the syntax errors corrected, four would become correct sentences with pronouns.

Diagnostics would be given in all cases of errors or system modification of the input. Diagnostics would also be given for each of the ambiguous input forms.

9.2.2 The Metric, More Fully: Evaluation Equivalents

The distance of ill-formed or well-formed incomplete input from well-formed complete input can be thought of as the number of evaluations required to produce a correction. Thus the use of more salient means of abbreviated reference is closer than the use of less salient means, and input requiring only minor corrections is closer than input requiring major corrections or input that is not correctable at all. Orderings were given within the major and minor corrections expressing decisions about the probability of a kind of change, and these also express variations in the distance from the original input to well-formed complete input.

This ordering was made dependent on the syntax of the encountered fragments; words with the phatic part of speech are deleted as part of the minor corrections, and variants of the

same root word are exchanged as one of the major corrections. It was also made dependent on the semantics, by requiring semantic evaluation before concluding on a particular modification. The ordering on evaluation attempts can thus be considered as a refinement of the metric in [Bradford 82a, 82b]. It further considers the methods of abbreviated reference within the same overall scheme.

No formal specification of this implied metric will be given. The part of this metric suitable to mathematical formalization (a Damerau-Levenshtein metric on words) was presented in [Bradford 82a, 82b], and to expand to the full metric suggested in this thesis would necessitate complete grammatical and semantic analysis. While this would be of theoretical value, it only serves here to give a motivation behind the ordering of evaluation attempts.

9.2.3 Future Work

There are many areas for future work in improving the habitability of interactive natural languages systems. Prospects for the treatment of some other kinds of anaphora were considered earlier. Even the complete processing of each of these types will not cover all kinds of abbreviated reference. Refinement of the rule relaxation methods and application to other words and parts of speech will increase the potential of this step for correcting ill-formed input. The grammar must be extended to accept all of the non-sentence non-whole arcs that constitute likely elliptical forms.

The search for better diagnostics should continue. As long as it is easy for users to exceed the boundaries of the set of inputs acceptable to the system, diagnosis will be necessary.

How to best diagnose the radical departures from these boundaries will depend on the experience and opinions of the user, but will always play a part in the habitability of a system.

New types of input modifications producing potential corrections can be considered within the larger system now, and how these may interact with other corrections would be seen by how they fit into the overall scheme. Their likelihood would determine their position in the ordering of evaluation attempts.

Some areas of well-formed, complete input are still problematic. Conjunction is one of these. These kinds of areas have not been considered in this thesis, as it is hoped that they will be treated within the grammar. This should be true at least for the commonly occurring cases. A pattern matching technique is proposed in [Kwasny 80] for treatment for several more general conjunction forms, but is described as still being incomplete.

Habitability remains to be considered from the points of view of specific application areas. This needs to take place in terms of both the well-formed forms of input specific to the problem context and in terms of the common exceptional forms. Protocol analysis of people working on the given task area is necessary in both cases.

9.2.4 Habitability: The Final Evaluation

Can the designers of natural language systems evaluate their work? In terms of abstract coverage of linguistic phenomena, theoretically yes. In terms of completeness with regard to an area of application, to an extent. In terms of

habitability? The system should at least pass the designer's test for habitability. The only real test for the completeness and habitability of a natural language system is with people trying to use the system. After gathering data from the interaction of users with the system, design and implementation for improved habitability will begin again. When this process reaches a "fixed point" system, one not changing with further protocol analysis, then it may be concluded that the system is (at least a local) maximum in terms of habitability. It is hoped that this thesis constitutes a major step in this direction.

Bibliography

Bigelow, R.B., N.R. Greenfeld, P. Szolovits, F.B. Thompson, Specialized Languages: An Applications Methodology, Proceedings of the National Computer Conference, 1973, 42, M49-M53.

Bowman, E., The Minor and Fragmentary Sentences of a Corpus of Spoken English, Bloomington, Indiana University, 1966.

Bradford, J.H., The Eta Interface - An Error Correcting Parser for Augmented Transition Networks, Ph.D. Thesis, Department of Computer Science, University of Waterloo, 1982(a).

Bradford, J.H., "A Metric Space Defined on English and its Relation to Error Correction," in COLING 82, North-Holland Publishing Company, 1982(b), pp. 43-48.

Bullwinkle, C.L., "The Semantic Component of PAL: The Personal Assistant Language Understanding Program," working paper 141, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, March 1977(a).

Bullwinkle, C.L., "Levels of Complexity in Discourse for Anaphora Disambiguation and Speech Act Interpretation," in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, August 1977(b).

Chapanis, A., R.N.Parrish, R.B.Ochsman and G.D.Weeks, "Studies in Interactive Communication: I. The Effects of Four Communication Modes on the Behavior of Teams During Cooperative Problem-Solving," Human Factors, Vol. 14, pp. 487-509, 1972.

Chapanis, A., "Interactive Human Communication," in Scientific American, April 1975, pp. 36-42.

Chapanis, A., R.N.Parrish, R.B.Ochsman and G.D.Weeks, "Studies in Interactive Communication: II. The Effects of Four Communication Modes on the Behavior of Teams During Cooperative Problem-Solving," Human Factors, Vol. 19, pp. 101-126, 1977.

Chomsky, N., Aspects of the Theory of Syntax, MIT Press, Cambridge, Massachusetts, 1965.

Codd, E.F., R.S. Arnold, J.M. Cadiou, C.L. Chang and N. Roussopoulos, "Rendezvous Version 1: An Experimental English Language Query Formulation System for Casual Users," Report RJ2144(29407), San Jose, CA., IBM Research Laboratory, 1978.

Cohen, P.R., S. Pertig and K. Starr, "Dependencies of Discourse Structure on the Modality of Communication: Telephone vs. Teletype," in Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics, University of Toronto, Toronto, Ontario, Canada, 1982, pp. 28-35.

Damerau, F.J., "A Technique for Computer Detection and Correction of Spelling Errors," in Communications of the ACM, Vol. 7, No. 3, 1964, pp. 171-176.

Damerau, F.J., "The Transformational Question Answering (TQA) System Operational Statistics," in Americal Journal of Computational Linguistics, Vol. 7, No. 1, pp. 30-42, 57-83, 1981.

Fries, C., The Structure of English, Harcourt, Brace and World, Inc., New York and Burlingame, 1952.

Grosz, B., "The Representation and Use of Focus in a System for Understanding Dialogs," in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, August 1977(a), pp. 67-76.

Grosz, B.J., The Representation and Use of Focus in Dialogue Understanding, Ph.D. Thesis, Department of Computer Science, University of California at Berkeley, June 1977(b).

Grosz, B.J., "Focusing in Dialogue," in Waltz, D.L., ed., TINLAP-2: Theoritical Issues in Natural Language Processing 2, University of Illinois at Urbana-Champaign, July 1978, pp. 96-103.

Hajicova, E., and J. Vrbova, "On the Role of the Hierarchy of Activation in the Process of Natural Language Understanding," in COLING 82, North-Holland Publishing Company, 1982, pp. 107-113.

Hall, P.A.V., and G.R. Dowling, "Approximate String Matching," in ACM Computing Surveys, Vol. 12, No. 4, 1980, pp. 381-402.

Harris, L.R., "User Oriented Data Base Query with the ROBOT Natural Language Query System," International Journal of Man-Machine Studies, 9, 1977, pp. 697-713.

Hayes, P.J., and R. Reddy, "An Anatomy of Graceful Interaction in Spoken and Written Man-Machine Communication," CMU-CS-79-144, Carnegie-Mellon University, Pittsburgh, PA., 1979.

Hayes, P.J., "Anaphora for Limited Domain Systems," in Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981(a), pp. 416-422.

Hayes, P.H., and G.V. Mouradian, "Flexible Parsing," in American Journal of Computational Linguistics, Vol. 7, No. 4, 1981(b), pp. 232-242.

Hendrix, G.G., E.D. Sacerdoti, D. Sagalowicz and J. Slocum, "Developing a Natural Language Interface to Complex Data," in ACM Transactions on Database Systems, Vol. 3, No. 2, June, 1978, pp. 105-147.

Hirst, G., Anaphora in Natural Language Understanding: A Survey, Lecture Notes in Computer Science, Springer-Verlag, New York, 1981.

Ho, T.P., Designing Knowledgeable Dialogue Systems, Ph.D. Thesis, Caltech, Pasadena, CA., 1983, forthcoming.

Kantor, R.N., The Management and Comprehension of Discourse Connection by Pronouns in English, Ph.D. Thesis, Department of Linguistics, Ohio State University, 1977.

Kaplan, S.J., "On the Difference Between Natural Language and High Level Query Languages," in ACM 78: Proceedings of the 1978 Annual Conference, Association for Computing Machinery, ACM, New York, 1978, Vol. 1, pp. 27-38.

Kaplan, S.J., Cooperative Responses from a Portable Natural Language Data Base Query System, Ph.D. Thesis, University of Pennsylvania, 1979.

Kasdan, L., and L. Brackett, The Empire Strikes Back, (screenplay), 1980.

Kelly, M.J., and A. Chapanis, "Limited Vocabulary Natural Language Dialogue," in International Journal of Man-Machine Studies, 1977, 9, pp. 479-501.

Kwasny, S.C., Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems, Ph.D. Thesis, Ohio State University, 1980.

Kwasny, S.C., and N.K. Sondheimer, "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems," American Journal of Computational Linguistics, Vol. 7, No. 2, 1981, pp. 99-108.

Nash-Webber, B.L., "Semantic Interpretation Revisited," Report 3335, (AI Report 48), Bolt Beranek and Newman Inc., Cambridge, Massachusetts, July 1976.

Nash-Webber, B.L., and R. Raymond, "Anaphora and Logical Form: On Formal Meaning Representations for Natural Language," in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, August 1977, pp. 121-131.

Malhotra, A., Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis, Ph.D. Thesis, MIT, MAC TR-146, 1975.

Malinowski, B., "The Problem of Meaning in Primitive Languages," in Ogden, C.J., and I.A.Richards, The Meaning of Meaning, Harcourt, Brace and Company, Inc., New York, 1946.

Marshall, C.C., "Peripheral Speech Elements," Unpublished Manuscript, Caltech, Pasadena, CA., 1976.

Marshall, C.C., An Analysis of Fragments in Human Dialogue, M.S. Thesis, Caltech, Pasadena, CA., 1980.

Mor, M., and A.S. Fraenkel, "A Hash Code Method for Detecting and Correcting Spelling Errors," in Communications of the ACM, Vol. 25, No. 12, 1982, pp. 935-938.

Morgan, H.L., "Spelling Correction in Systems Programs," in Communications of the ACM, Vol. 13, No. 2, 1970, pp. 90-94.

Peterson, J.L., Computer Programs for Spelling Correction: An Experiment in Program Design, Lecture Notes in Computer Science, Springer-Verlag, New York, 1980.

Osborne, D., The Andromedans, InterVarsity Press, Dowers Grove, Illinois, 1977.

Riseman, E.M., and A.R. Hanson, "A Contextual Post-Processing System for Error-Correction Using Binary N-Grams," in IEEE Transactions on Computers, C-23, 5, 1974, pp. 480-493.

Roach, K., Pronouns, M.S. Thesis, Caltech, Pasadena, CA., 1980.

Sgall, P., and E. Hajicova, "Focus on Focus," in Prague Bulletin of Mathematical Linguistics, 28(1977) pp. 5-51, and 29(1978) pp. 22-41.

Shneiderman, B., "Natural vs. Precise Concise Language for Human Operation of Computers: Research Issues and Experimental Approaches," in Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, 1980, pp. 139-141.

Shneiderman, B., "Designing Computer System Messages," in Communications of the ACM, Vol. 25, No. 9, 1982, pp. 610-611.

Sidner, C.L., "A Progress Report on the Discourse and Reference Components of PAL," in Proceedings of the Second National Conference: Canadian Society for Computational Studies of Intelligence, Toronto, July 1978(a), pp. 206-213.

Sidner, C.L., "The Use of Focus as a Tool for the Disambiguation of Definite Noun Phrases," in Waltz, D.L., ed., TINLAP-2: Theoretical Issues in Natural Language Processing 2, University of Illinois at Urbana-Champaign, July 1978(b), pp. 86-95.

Sidner, C.L., Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1979.

Sidner, C.L., "Focus for Interpretation of Pronouns," in American Journal of Computational Linguistics, v. 7, no. 4, 1981, pp. 217-231.

Smith, J., "English: A Clear Case of Confusion," in The Los Angeles Times, May 1981, part IV, p. 1.

Sondheimer, N.K., and R.M. Weischedel, "A Rule-Based Approach to Ill-Formed Input," in COLING 80: Proceedings of the 8th International Conference on Computational Linguistics, Tokyo, October 1980, pp. 46-53.

Tennant, H.R., Evaluation of Natural Language Processors, Ph.D. Thesis, University of Illinois, Urbana, IL., 1980.

Thompson, F.B., "English for the Computer," Proceedings of the Fall Joint Computer Conference 29, pp. 349-356, 1966.

Thompson, F.B., and B.H. Thompson, "Practical Natural Language Processing: The REL System as Prototype," in Advances in Computers, Vol. 13, New York, Academic Press, 1975.

Thompson, B.H., REL English for the User, REL Project Report, Caltech, Pasadena, CA., 1978.

Thompson, B.H., "Linguistic Analysis of Natural Language Communication with Computers," COLING 80: Proceedings of the 8th International Conference on Computational Linguistics, Tokyo, October 1980, pp. 190-201.

Thompson, B.H., F.B. Thompson, and T.P. Ho, "Knowledgeable Contexts for User Interaction," Technical Memorandum 5051:TM:82, Caltech, Pasadena, CA., 1982(a), to appear in Proceedings of the National Computer Conference, Anaheim, CA., May 1983.

Thompson, B.H., and F.B. Thompson, "Introducing ASK, A Simple Knowledgeable System," Technical Memorandum 5054:TM:82, Caltech, Pasadena, CA., 1982(b), also in Proceedings of the Conference on Applied Natural Language Processing, Santa Monica, CA., February 1983, pp. 17-24.

Thompson, B.H., "Linguistic Analysis of Pronouns in Interactive Discourse," Technical Memorandum, Caltech, Pasadena, CA., 1983, forthcoming.

Trawick, D.J., Towards More Natural Diagnostics for Interactive Natural Language Applications, Unpublished M.S. Thesis, Caltech, Pasadena, CA., 1979.

Walsh, B., and D. DaGradi, Mary Poppins, (screenplay), 1964.

Watt, W.C., "Habitability," in American Documentation, Vol. 19, No. 3, 1968, pp. 338-351.

Webber, B.L., A Formal Approach to Discourse Anaphora, Ph.D. Thesis, Harvard University, 1978(a).

Webber, B.L., "Description Formation and Discourse Model Synthesis," in Waltz, D.L., ed., TINLAP-2: Theoretical Issues in Natural Language Processing 2, University of Illinois at Urbana-Champaign, July 1978(b), pp. 42-50.

Weischedel, R.M., Please Re-Phrase, TR #77/1, Department of Statistics and Computer Science, Newark, University of Delaware, 1977.

Weischedel, R.M., W.M. Voge, and M. James, "An Artificial Intelligence Approach to Language Instruction," in Artificial Intelligence, 10, 1978, pp. 225-240.

Weischedel, R.M., and J. Black, "Responding Intelligently to Unparsable Inputs," American Journal of Computational Linguistics, Vol. 6, No. 2, 1980, pp. 97-109.

Weischedel, R.M., and N.K. Sondheimer, A Framework for Processing Ill-Formed Input, Department of Computer and Information Sciences, University of Delaware, 1981.

Winograd, T., Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, MAC TR-84, Ph.D. Thesis, MIT, 1971.

Woods, W.A., "Progress in Natural Language Understanding -- An Application to Lunar Geology," in Proceedings of the National Computer Conference, Montvale, N.J., AFIPS Press, 1973.

Yu, K.I., Communicative Databases, Ph.D. Thesis, Caltech, Pasadena, CA., 1980.

Appendix 1: Handouts for Experiments

The following pages are the handouts given to the subjects for the protocol analysis experiments. Person 1 and Person A for the face-to-face and terminal-to-terminal modes are given first. The handout for the human-to-computer mode follows these.

(A1.1 Face-to-Face and Terminal-to-Terminal Handout)

FOR PERSON 1

Data about the decks of the Alamo.

deck name	min. cl.	sq. ft.	hatch	primary use
magazine 3-108-2-m	85	200	36x30	ammunition
magazine 3-63-2-m	83	367	30x30	ammunition
magazine 3-84-2-m	74	312	36x30	ammunition
pyrotechnic locker	40	36	---	pyrotechnics
super deck	---	3132	---	vehicles/pallets
well deck	138	18573	---	vehicles/pallets
mezzanine deck	138	5960	---	vehicles/pallets
special weapons magazine	60	100	---	special weapons

Cargo items: name and number to be stowed.

chg demo	18	Loaderscptrac	1
conwire	84	lub oil	2
ctg 105smk	8	MCI	132
ctg 105wp	9	mine M16	14
ctg 105he(1)	32	nw1	2
ctg 105he(2)	48	nw2	12
ctg 106ape	12	nw3	2
ctg 40he	18	post 2ft	14
ctg 50	30	recvryveh	3
ctg 5.56	18	rollertowshpft	1
ctg 60	10	sandbag	30
ctg 7.62	35	semitrlr 20tstk	1
culvert	5	smoke pot	4
flare sur	1	tank 90mm M48	1
fuel oil	4	trac M64 w/rup	5
fuze M564	48	trk crash MB-5	2
fuze smk	1	trk fire pump	1
gen set	1	trkfklft 6000rt	3
gren imc	2	trk trac M260	1
gren smk	9	trk trac M52 W	5
how 155mm tow(1)	5	trk util M676	2
how 155mm tow(2)	6	trk util F2W	6
how 8in SP M110	10	water	48
ign	1	wire 350	6
LVTE1	4	LVTP5A1 w/dzr	8
LVTP5A1	2		

FOR PERSON A

Cargo items, listed by class, with: name, l, w, h, wt, area.
(l, w, h in inches; area in sq ft)

SUBSISTENCE (class i):
MCI, 48, 40, 48, 1344, 14
water, 48, 40, 24, 910, 14

FORTIFICATION (class iv):
conwire, 48, 40, 30, 350, 14
wire 350, 48, 40, 41, 1537, 14
culvert, 48, 40, 39, 945, 14
post 2ft, 50, 42, 13, 1370, 15
sandbag, 48, 40, 41, 650, 14

POL (class iii):
lub oil SAE 50, 32, 18, 24, 120, 4
fuel oil, 40, 40, 50, 1500, 12

GENERAL SUPPLIES (class ii):
gen set PU670G, 48, 48, 50, 1500, 16

AMMUNITION (class v):
ctg 5.56, 36, 30, 21, 633, 8
ign, 35, 25, 19, 140, 6
ctg 7.62, 36, 20, 17, 217, 5
ctg 50, 36, 30, 21, 885, 8

PYROTECHNICS (class vp):
ctg 105smk, 30, 30, 29, 885, 7
gren smk, 30, 30, 13, 182, 7
gren imc, 48, 40, 14, 274, 14
smoke pot, 36, 30, 34, 390, 8
ctg 105wp, 30, 20, 50, 1080, 5
flare sur, 48, 40, 31, 532, 14

DEMOLITIONS (class vd):
fuze smk, 11, 8, 6, 10, 1
ctg 106ape, 30, 25, 30, 675, 6
fuze m564, 32, 28, 15, 230, 7
chg dome, 32, 20, 16, 158, 5
ctg 105he(1), 36, 30, 29, 829, 8
ctg 105he(2), 25, 21, 29, 425, 4
ctg 40he, 25, 25, 27, 354, 4
ctg 60he, 30, 30, 35, 458, 7
mine m16, 36, 25, 30, 792, 7

SPECIAL WEAPONS (class vsw):
nw1, 48, 40, 20, 1500, 14
nw2, 47, 45, 17, 1200, 15
nw3, 48, 40 21, 1267, 14

VEHICLES (class vii):
LVTE1, 474, 151, 129, 96200, 497
LVTP5A1, 356, 141, 115, 82500, 349
tank 90mm M48, 292, 144, 123, 104000, 292
trk trac M260, 262, 96, 99, 17160, 175
how 8in SP M110, 293, 124, 111, 58500, 252
how 155mm tow(1), 293, 97, 93, 12800, 197
how 155mm tow(2), 288, 96, 81, 12700, 192
LVTP5A1 w/dzr, 402, 140, 114, 88500, 413
recvryveh M51, 399, 143, 129, 120000, 396
trk util M676, 180, 78, 87, 4587, 98
trk crash MB-5, 292, 98, 128, 12100, 199
trk fire pump, 267, 91, 104, 15380, 169
trkfklft 6000rt, 204, 86, 124, 18000, 122
trk trac M52 W, 274, 98, 103, 19130, 187
trlr util F2W, 122, 96, 04, 900, 02
trac M64 w/rup, 257, 129, 116, 49300, 231
semitrlr 20tstk, 480, 96, 55, 12250, 320
rolertowshpft, 198, 97, 88, 20750, 134
loaderscp trac, 313, 84, 106, 30708, 183

(A1.2 Human-to-Computer Mode Handout:)

Cargo to be loaded onto the Alamo

item name	number
MCI	28
water	32
conwire	27
wire 350	6
culvert	5
post 2ft	14
sandbag	30
lub oil SAE50	2
fuel oil	4
gen set PU670G	1
LVTE1	4
LVTP5A1	2
tank 90mm M48	1
trk trac M260	1
how 8in SP M110	10
how 155mm tow(1)	5
how 155mm tow(2)	6
LVTP5A1 w/dzr	8
recvryveh.	3
trk util M676	2
trk crash MB-5	2
trk fire pump	1
trkfklft 6000rt	3
trk trac M52 W	5
trlr util F2W	6
trac M64 w/rup	5
semitrlr 20tstk	1
rollertowshpft	1
ctg 5.56	18
ign	1
ctg 7.62	35
ctg 50	30
ctg 105smk	8
gren smk	9
gren imc	2
smoke pot	4
ctg 105wp	9
flare sur	1
nw1	2
mine m16	14
ctg 60he	10
ctg 40he	18
ctg 105he(2)	20
ctg 105he(1)	32
chg demo	18
fuze M564	28
ctg 106ape	12
fuze smk	1

meanings of classes:

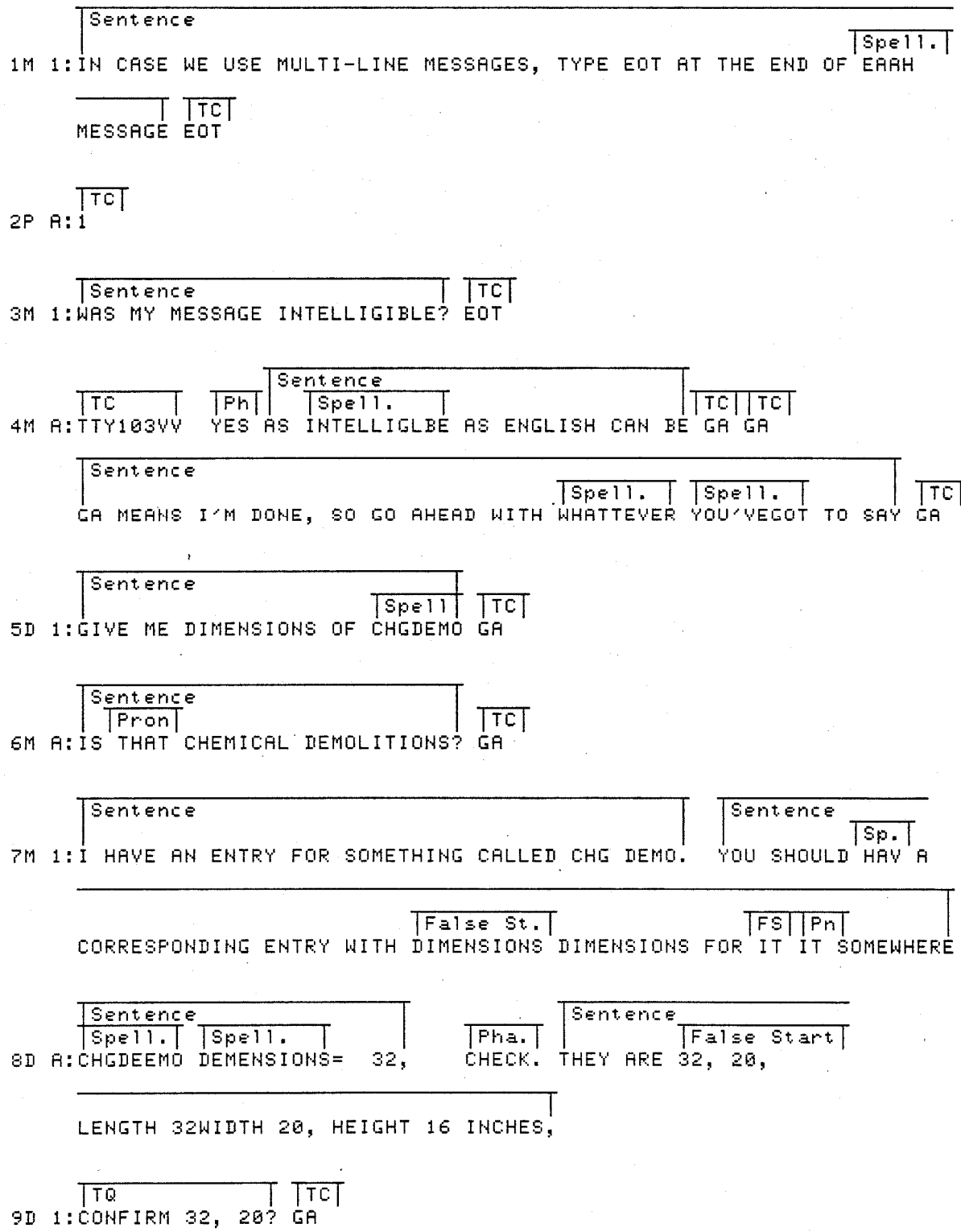
class i = subsistence
class ii = general supplies
class iii = POL
class iv = fortification
(The above 4 are considered "palletized cargo")
class v = ammunition
class vp = pyrotechnics
class vd = demolitions
class vsw = special weapons
class vii = vehicles

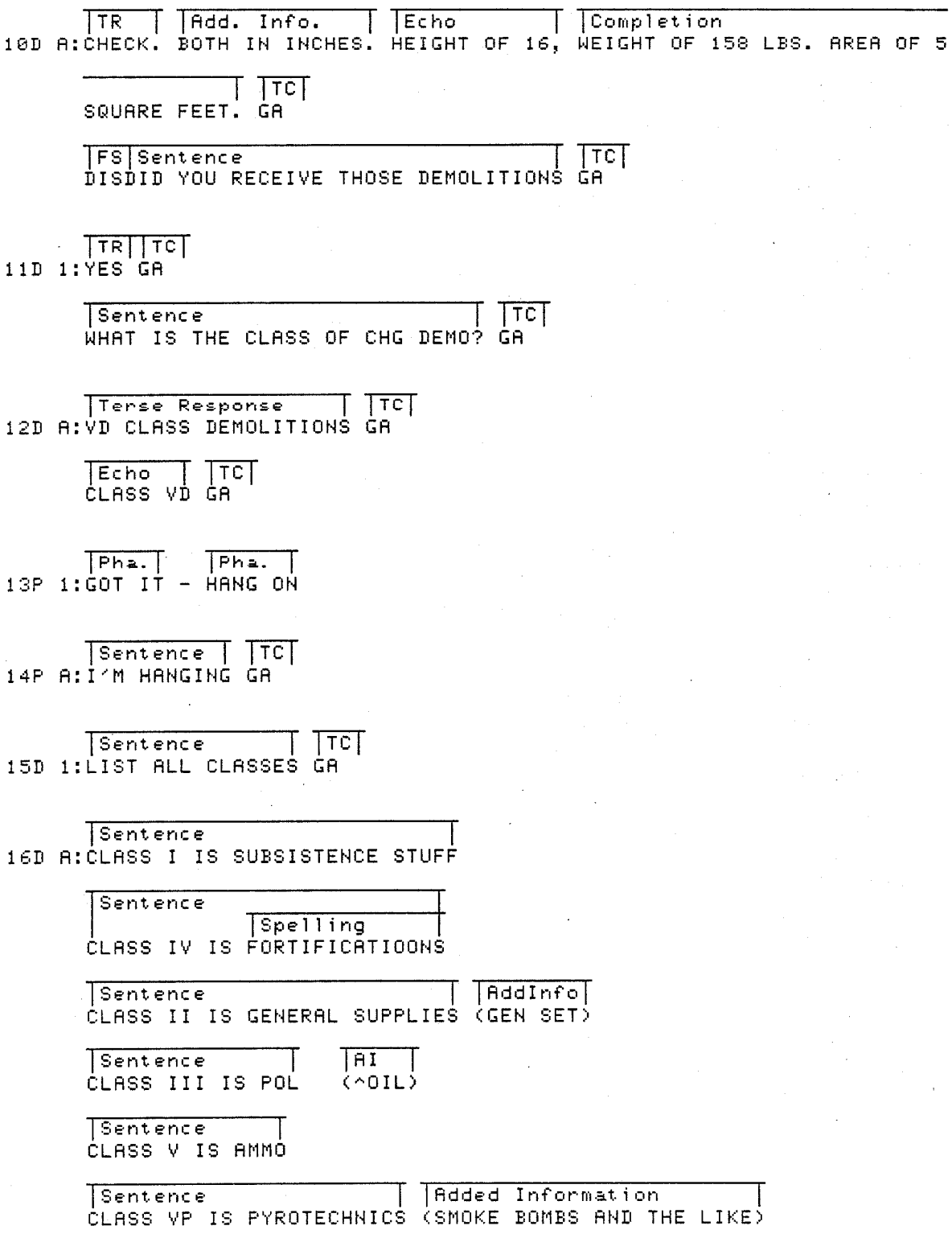
Appendix 2: Annotated Terminal-to-Terminal Protocol

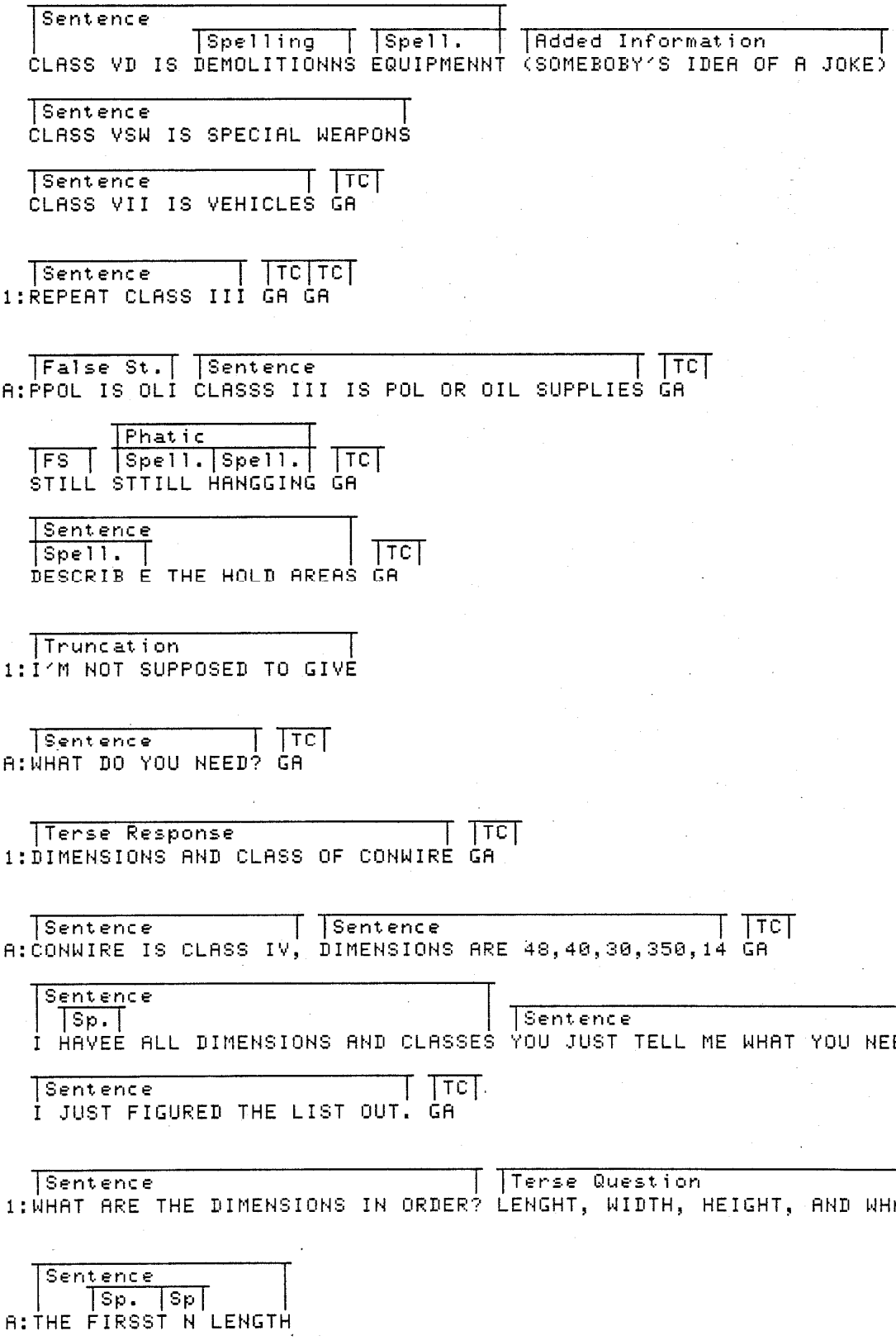
One of the terminal-to-terminal protocols taken in the experimental portion of the work is presented here with full analysis. The types of fragments are labeled in the protocol. Abbreviations follow the conventions of chapter 1. In addition, TC stands for termination character. This is a type of phatic, but was not counted in the analysis. TCs were used to indicate an end of transmission, a convention which all subjects in the terminal-to-terminal mode developed and subsequently violated. The first number indicates the message number. The letters P, M, D, or X come after the message number and correspond to message types of phatic, meta, data, or mixed. The 1 or A preceding the actual message is the speaker, either person 1 or person A.

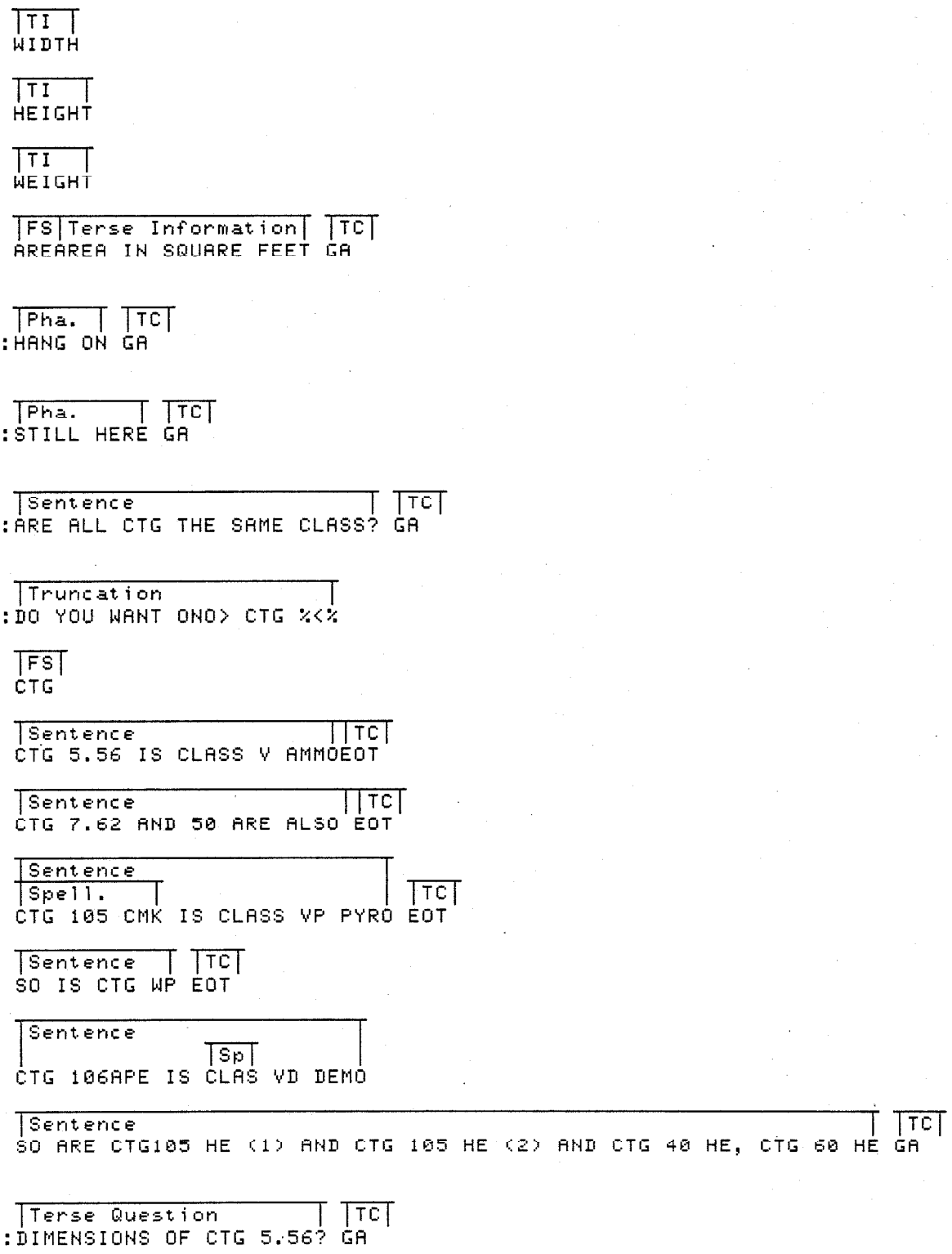
The statistical analysis, begining on page 150, gives the distribution of the individual fragment type's lengths. This analysis is for the annotated protocol. There is a graph for each category other than phatic and phatic connector. In the graphs, the vertical axis indicates the length in words of the fragments. The horizontal axis indicates the number of occurrences of fragments with that length. The mean and standard deviation are of the fragment's length. The phatics and phatic connectors are just listed along with the number of occurrences of each.

A2.1 Terminal-to-Terminal Protocol (DT T-T)




TR
Add. Info.
Echo
Completion
10D A:CHECK. BOTH IN INCHES. HEIGHT OF 16, WEIGHT OF 158 LBS. AREA OF 5
TC
SQUARE FEET. GA
FS
Sentence
TC
DISDID YOU RECEIVE THOSE DEMOLITIONS GA
TR
TC
11D 1:YES GA
Sentence
TC
WHAT IS THE CLASS OF CHG DEMO? GA
Terse Response
TC
12D A:VD CLASS DEMOLITIONS GA
Echo
TC
CLASS VD GA
Pha.
Pha.
13P 1:GOT IT - HANG ON
Sentence
TC
14P A:I'M HANGING GA
Sentence
TC
15D 1:LIST ALL CLASSES GA
Sentence
16D A:CLASS I IS SUBSISTENCE STUFF
Sentence
Spelling
CLASS IV IS FORTIFICATIOONS
Sentence
AddInfo
CLASS II IS GENERAL SUPPLIES (GEN SET)
Sentence
AI
CLASS III IS POL (^OIL)
Sentence
CLASS V IS AMMO
Sentence
Added Information
CLASS VP IS PYROTECHNICS (SMOKE BOMBS AND THE LIKE)


Sentence
Spelling
Spell.
Added Information
CLASS VD IS DEMOLITIONNS EQUIPMENNT (SOMEBOBY'S IDEA OF A JOKE)
Sentence
CLASS VSW IS SPECIAL WEAPONS
Sentence
TC
CLASS VII IS VEHICLES GA
Sentence
TC TC
17D 1:REPEAT CLASS III GA GA
False St.
Sentence
TC
18X A:PPOL IS OLI CLASSS III IS POL OR OIL SUPPLIES GA
Phatic
FS
Spell. Spell.
TC
STILL STTILL HANGGING GA
Sentence
Spell.
TC
DESCRIB E THE HOLD AREAS GA
Truncation
19M 1:I'M NOT SUPPOSED TO GIVE
Sentence
TC
20M A:WHAT DO YOU NEED? GA
Terse Response
TC
21D 1:DIMENSIONS AND CLASS OF CONWIRE GA
Sentence
Sentence
TC
22X A:CONWIRE IS CLASS IV, DIMENSIONS ARE 48,40,30,350,14 GA
Sentence
Sp.
Sentence
I HAVEE ALL DIMENSIONS AND CLASSES YOU JUST TELL ME WHAT YOU NEED.
Sentence
TC
I JUST FIGURED THE LIST OUT. GA
Sentence
Terse Question
TC
23D 1:WHAT ARE THE DIMENSIONS IN ORDER? LENGHT, WIDTH, HEIGHT, AND WHAT? GA
Sentence
Sp. Sp
24D A:THE FIRSST N LENGTH


TI
WIDTH
TI
HEIGHT
TI
WEIGHT
FS Terse Information TC
AREAREA IN SQUARE FEET GA
Pha. TC
25P 1:HANG ON GA
Pha. TC
26P A:STILL HERE GA
Sentence TC
27D 1:ARE ALL CTG THE SAME CLASS? GA
Truncation
28D A:DO YOU WANT ONO> CTG %<%
FS
CTG
Sentence TC
CTG 5.56 IS CLASS V AMMOEOT
Sentence TC
CTG 7.62 AND 50 ARE ALSO EOT
Sentence
Spell. TC
CTG 105 CMK IS CLASS VP PYRO EOT
Sentence TC
SO IS CTG WP EOT
Sentence
Sp
CTG 106APE IS CLAS VD DEMO
Sentence TC
SO ARE CTG105 HE (1) AND CTG 105 HE (2) AND CTG 40 HE, CTG 60 HE GA
Terse Question TC
29D 1:DIMENSIONS OF CTG 5.56? GA

|Terse Response | |TC|
30D A:36,36,30,21,633,8 GA

|Terse Question | |TC|
31D 1:36,36,30,21,633,8? GA

|TR||TC|
32D A:YES GA

|Phatic |
WAIT A MINUTE

|Sentence | |Sentence | |TC|
33D 1:I ONLY EXPECTED 5 MEASUREMENTS. WHAT ARE THOSE NUMBERS? GA

|Tr. Res. | |TC|
34D A:ONLY ONE 36. GA

|Pha.| |Pha. |
35D 1:GOT IT - HANG ON

|Terse Question | |TC|
DIMENSIONS OF CTG 7.62? GA

|Terse Res. | |TC|
36D A:36,20,17,217,5 GA

|Pha. |
37D 1:HANG ON

|Terse Question | |TC|
DIMENSIONS OF CTG 50? GA

|Terse Res. | |TC|
38D A:36,30,21,885,8 GA

|Pha. |
39D 1:HANG ON

|Sentence | |TC|
WHAT ELSE IS CLASS V? GA

|TC||TR||TC|
40D A:77+ IGN GA

```
      |Terse Qu.| |TC|
41D 1:DIMENSIONS? GA

      |TC|  |Terse Res.  | |TC|
42D A:700+  35,25,19,140,6 GA

      |Phatic                  |
      ::::::::::::::::::::::::

      |Sentence                                    | |Trunc.|
43D 1:CONFIRM IGN DIMENSIONS 35,25,19,140,  I REPEAT

      |TC| |Trun|
44D A:700+ 35,25,

      |Sentence                                    | |TC|
45D 1:CONFIRM IGN DIMENSIONS 35,25,19,140,6? GA

      |TR   | |TC|
46  A:CONFIRM GA

      |Sentence        |
47M 1:WHAT MEANS 700+  ?

      |Terse Response                          | |TC|
48M A:SAME AS YOUR TTY 102 EXCEPT FOR ME GA

      |Pha. |
49D 1:HANG ON

      |Sentence          | |TC|
      WHAT ELSE IS CLASS V? GA

      |TC|
      GA

      |TC| |TR   | |TC|
50D A:700+ NOTHING GA

      |Terse Question   | |TC|
51D 1:HOW ABOUT CLASS VP? GA

      |False Start|  |TC| |Sentence                  |
52D A:CLASS VP INCL  700+  THIS IS A LIST OF CLASS VP
```

|TI|
GREN SMK

|TI|
GREN IMC

|TI|
SMOKE POT

|TI| |PC||Terse Information| |Ph| |Sentence |Sp| | |TC|
FLARE SUR AND THE CTG'S ALREADY MENTIONED YES, IT SO SMOKE POT. GA

|Terse Question| |TC|
53D 1:DIMENSIONS CTG 105 SMK? GA

|TC| |Terse Res.| |TC|
54D A:700+ 30,30,29,885,7 GA

|Sentence|
55M 1:LEAVE OUT WEIGHTS

|TC||TR||TC|
56M A:700+ OK GA

|Terse Question| |TC|
57D 1:DIMENSIONS OF CTG 105 WP? GA

|Ter.Res.| |TC|
58D A:30,20,50,5 GA

|Terse Question| |TC|
59D 1:DIMENSIONS OF GREN SMK? GA

|TC| |FS| |Ter.Res.| |TC|
60D A:700+ 30,30,1 30,30,13,7 GA

|Ter.Ques.| |TC|
61D 1:30,30,13,7? GA

|TC||TR| |TC|
62D A:700+ YES GA

|Terse Question| |TC|
63D 1:DIMENSIONS OF GREN IMC? GA

```
      |TC| |Ter.Res. | |TC|
64D A:'00+ 48,40,14,14 GA

      |Terse Question     | |TC|
65D 1:SQ. FT. OF SMOKE POT? GA

      |TC ||TR||TC|
66D A:700 + 8  GA

      |Ter.Ques.  | |TC|
67D 1:OF FLARE SUR? GA

      |TC||TR||TC|
68D A:700: 14  GA

      |Ph.| |Phatic    | |Sentence                                                | |TC|
69P 1:OKAY, YOU TURKEY, I THINK THEY'RE TRYING TO TELL ME WE'RE DONE GA

      |TC | |Sentence       | |TC|
70P A:7700: THAT'S FINE, HERE GA

      |Phatic |
71P 1:GOOD BYE!

      |Sentence
      |TC||                  |FS | |FS  ||FS||FS  |           | |TC|
72P A:700:DO YOU WANT SOME TRKET TURKET TY TURKET TRY TURKEY? GA
```

A2.2 Statistical Analysis

messages DT T-T

mean= 7.875 std dev= 9.881 total= 72.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
 1|====-
 2|==-
 3|===-
 4|=======
 5|====
 6|===
 7|=-
 8|==
 9|=
10|-
11|-
12|=
  ~
15|-
16|
17|=
  ~
20|-
21|
22|=
  ~
28|-
29|
30|-
  ~
53|-
54|
55|-
```

sentences DT T-T

mean= 5.852 std dev= 2.765 total= 54.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
 2|-
 3|==
 4|========-
 5|====-
 6|====-
 7|----
 8|=
 9|=-
10|
11|-
  ~
15|-
  ~
18|-
```

completion DT T-T

mean= 9.000 std dev= 0.000 total= 1.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
 9|-
```

truncation DT T-T

mean= 3.750 std dev= 1.785 total= 4.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
 2|=
  ~
 5|-
 6|-
```

false start DT T-T

mean= 1.500 std dev= 0.824 total= 14.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
 1|=====
 2|-
 3|=-
```

echo DT T-T

mean= 2.500 std dev= 0.500 total= 2.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
  2 |-
  3 |-
```

terse question DT T-T

mean= 4.000 std dev= 1.095 total= 15.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
  1 |-
  2 |
  3 |=
  4 |====
  5 |=-
  6 |-
```

terse reply DT T-T

mean= 3.045 std dev= 2.099 total= 22.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
  1 |=====
  2 |
  3 |=
  4 |=-
  5 |==-
  6 |-
  7 |
  8 |-
```

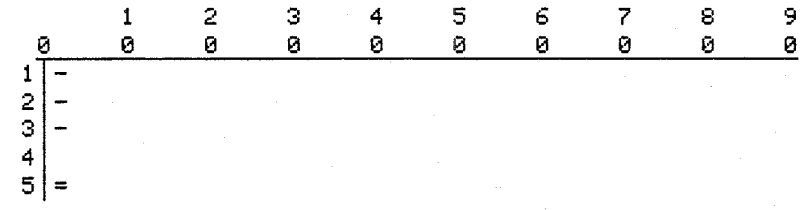
terse information DT T-T

mean= 2.111 std dev= 1.100 total= 9.0

```
      1    2    3    4    5    6    7    8    9
  0   0    0    0    0    0    0    0    0    0
  1 |=-
  2 |==
  3 |
  4 |=
```

added information DT T-T

mean= 3.200 std dev= 1.600 total= 5.0



phatics DT T-T word, (number of occurrences) total 18.0

hang on (6)	check (1)	wait a minute (1)
OK (2)	yes (1)	(symbols) (1)
still hanging (2)	you turkey (1)	
got it (2)	good bye (1)	

phatic connectors DT T-T total 2.0

so (1) and (1)

fragment total 93

Appendix 3: Pronoun List

This is the list of pronouns as they appear in the lexical rules. Explanation of the features is given in the table preceeding the lexical values. All of these words have the part of speech "pronoun" as at least one of their parts of speech.

Features:

anf	animate feature
nanf	neuter animate feature (animate not checked)
gnf	gender feature (indicates female)
neu	neuter feature (gender not checked)
pof	possessive feature
ref	reflexive feature
plf	plural feature
nplf	neuter plural feature (plural not checked)

<pronoun>	<anf>	<nanf>	<gnf>	<neu>	<pof>	<ref>	<plf>	<nplf>
"he"	+	-	-	-	-	-	-	-
"she"	+	-	+	-	-	-	-	-
"s/he"	+	-	-	+	-	-	-	-
"his"	+	-	-	-	+	-	-	-
"him"	+	-	-	-	-	-	-	-
"himself"	+	-	-	-	-	+	-	-
"her"	+	-	+	-	+	-	-	-
"her"	+	-	+	-	-	-	-	-
"herself"	+	-	+	-	-	+	-	-
"him/herself"	+	-	-	+	-	+	-	-
"him/her"	+	-	-	+	-	-	-	-
"his/her"	+	-	-	+	+	-	-	-
"it"	-	+	-	+	-	-	-	-
"its"	-	+	-	+	+	-	-	-
"itself"	-	+	-	+	-	+	-	-
"they"	-	+	-	+	-	-	+	+
"their"	-	+	-	+	+	-	+	+
"them"	-	+	-	+	-	-	+	+
"themselves"	-	+	-	+	-	+	+	+
"one"	-	+	-	+	-	-	-	-
"this one"	-	+	-	+	-	-	-	-
"that one"	-	+	-	+	-	-	-	-
"those"	-	+	-	+	-	-	-	+
"these"	-	+	-	+	-	-	+	+
"those"	-	+	-	+	-	-	+	+
"that of"	-	+	-	+	-	-	-	+
"those of"	-	+	-	+	-	-	-	+
"same of"	-	+	-	+	-	-	-	+